Graphics in Go

@peterhellberg

Interfaces!

image.Image

```
// Image is a finite rectangular
// grid of color.Color values taken from a color model.
type Image interface {
    // ColorModel returns the Image's color model.
    ColorModel() color.Model
    // Bounds returns the domain for which At can return non-zero color.
    // The bounds do not necessarily contain the point (0, 0).
    Bounds() Rectangle
    // At returns the color of the pixel at (x, y).
    At(x, y int) color.Color
}
```

color.Model

```
// Model can convert any Color to one from its own color model.
// The conversion may be lossy.
type Model interface {
  Convert(c Color) Color
}
```

color.Color

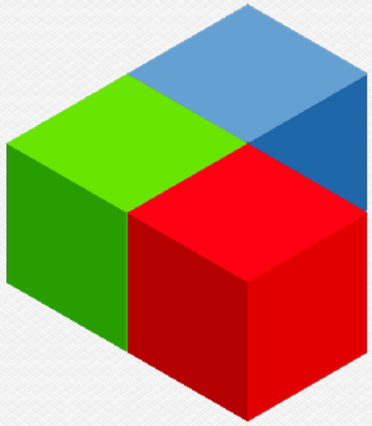
```
// Color can convert itself to alpha-premultiplied 16-bits per channel RGBA.
// The conversion may be lossy.
type Color interface {
    // RGBA returns the alpha-premultiplied red, green, blue and alpha values
    // for the color. Each value ranges within [0, 0xffff], but is represented
    // by a uint32 so that multiplying by a blend factor up to 0xffff will not
    // overflow.
    //
    // An alpha-premultiplied color component c has been scaled by alpha (a),
    // so has valid values 0 <= c <= a.
    RGBA() (r, g, b, a uint32)
}
```

draw.Image

```
// Image is an image.Image with a Set method to
change a single pixel.
type Image interface {
  image.Image
  Set(x, y int, c color.Color)
}
```

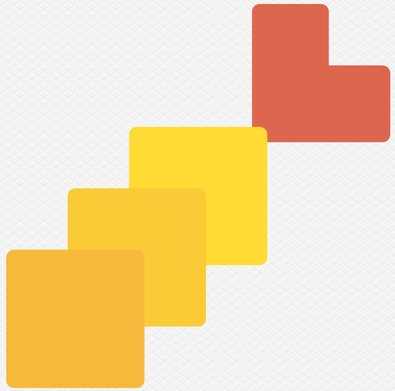
What about the elephant in the room? (Shaders)

Packages?



gfx

go get github.com/peterhellberg/gfx



ebiten

go get github.com/hajimehoshi/ebiten

pixel

go get github.com/faiface/pixel

#gamedev

slack.gopher.se


29
21

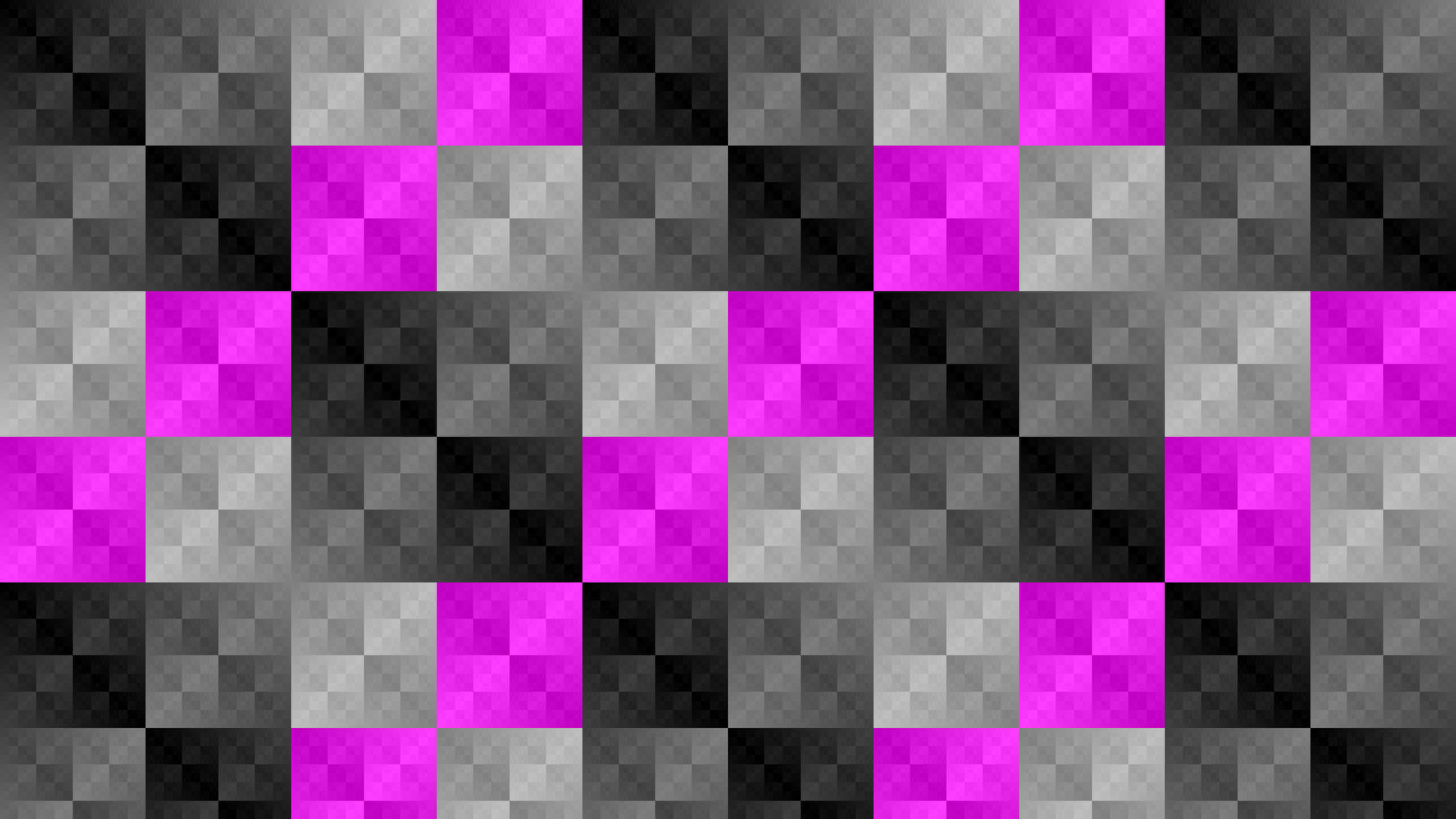
Experiments

The XOR texture

lodev.org/cgtutor/xortexture.html

The XOR texture

```
package main

import "github.com/peterhellberg/gfx"

func main() {
  m := gfx.NewImage(640, 360)

  gfx.EachPixel(m.Bounds(), func(x, y int) {
    c := uint8(x ^ y)

    m.Set(x, y, gfx.ColorNRGBA(c, c%192, c, 255))
  })

  gfx.SavePNG("xor.png", gfx.NewScaledImage(m, 4))
}
```

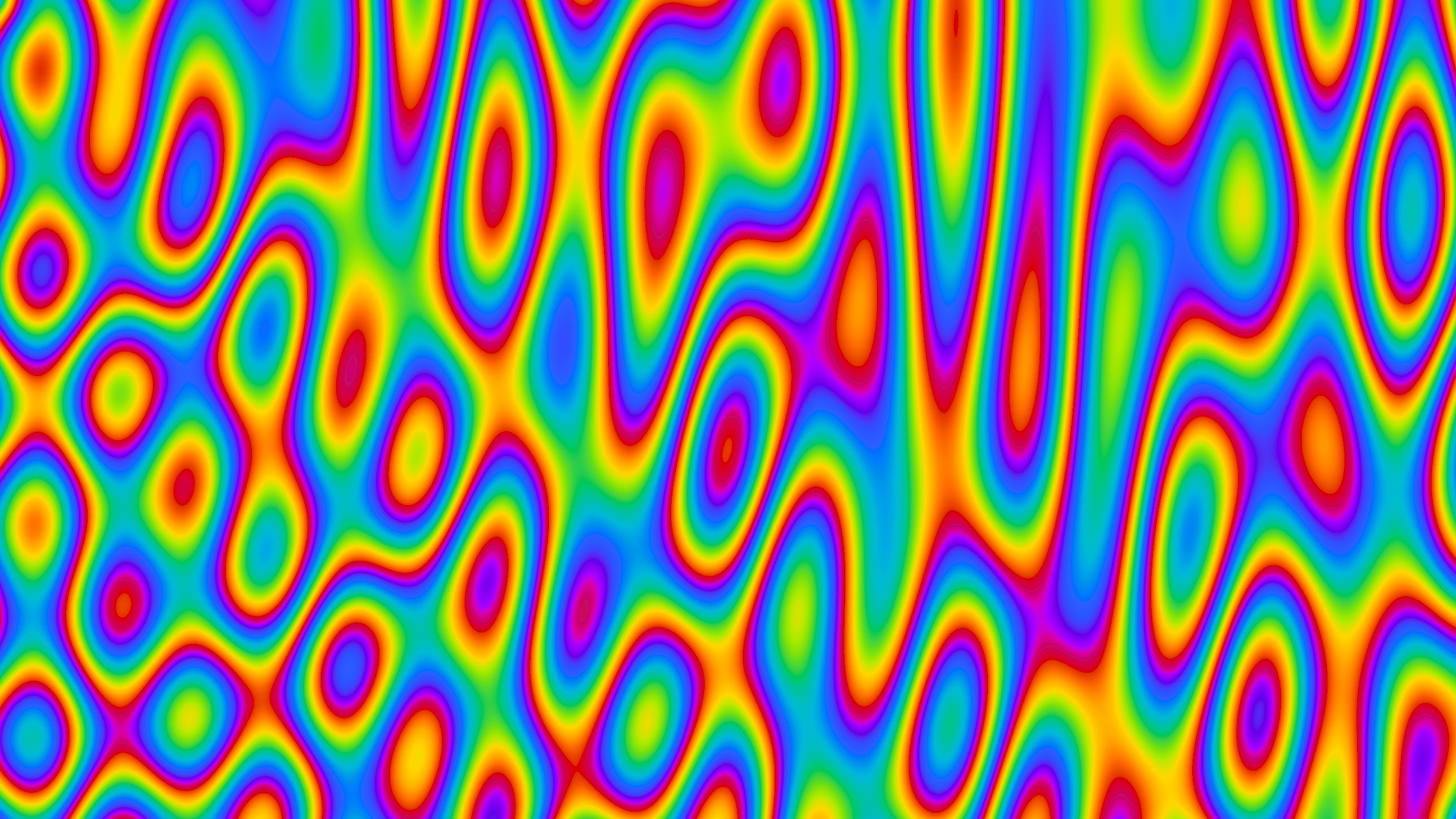


Plasma

lodev.org/cgtutor/plasma.html

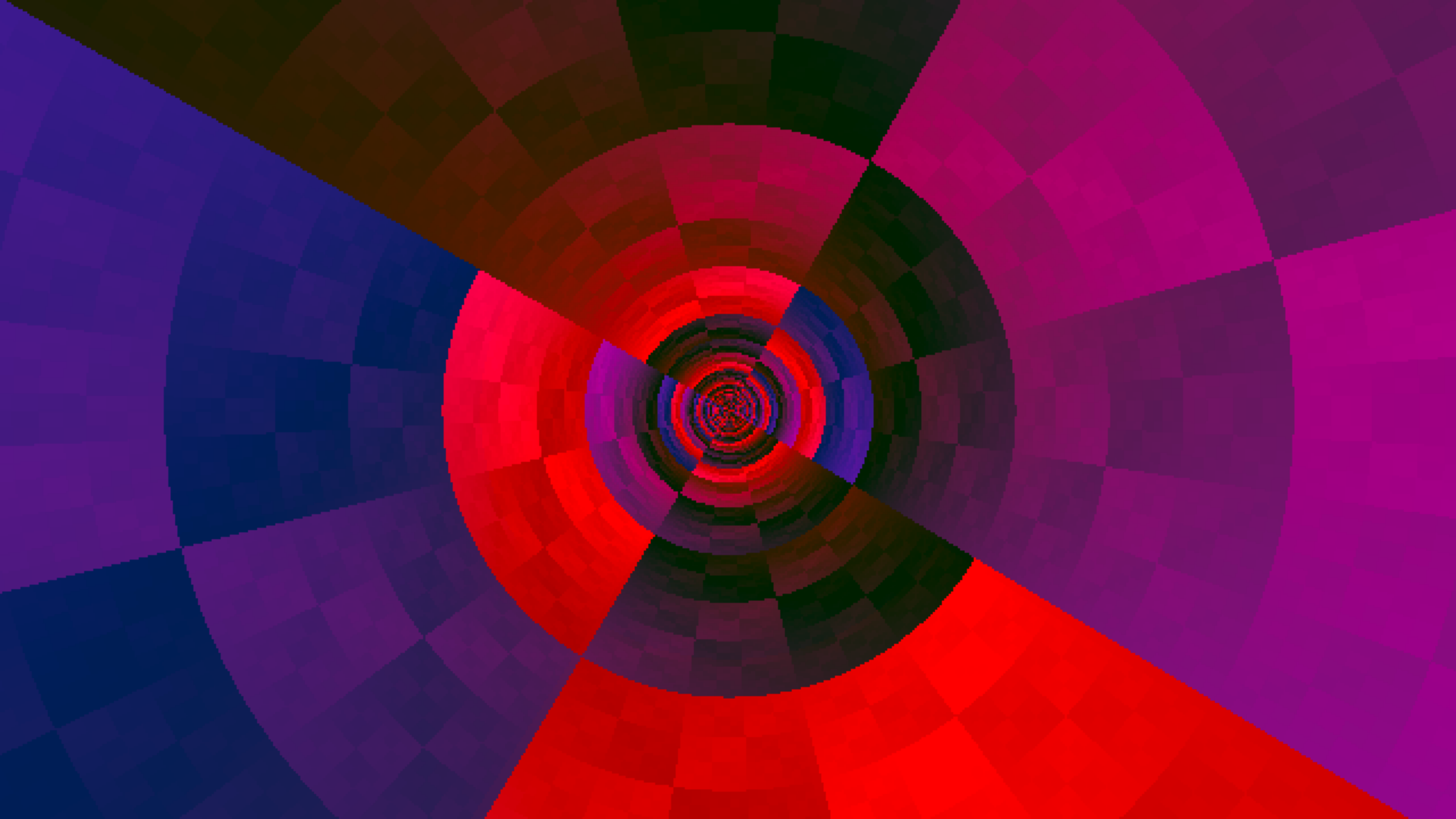
The Plasma effect

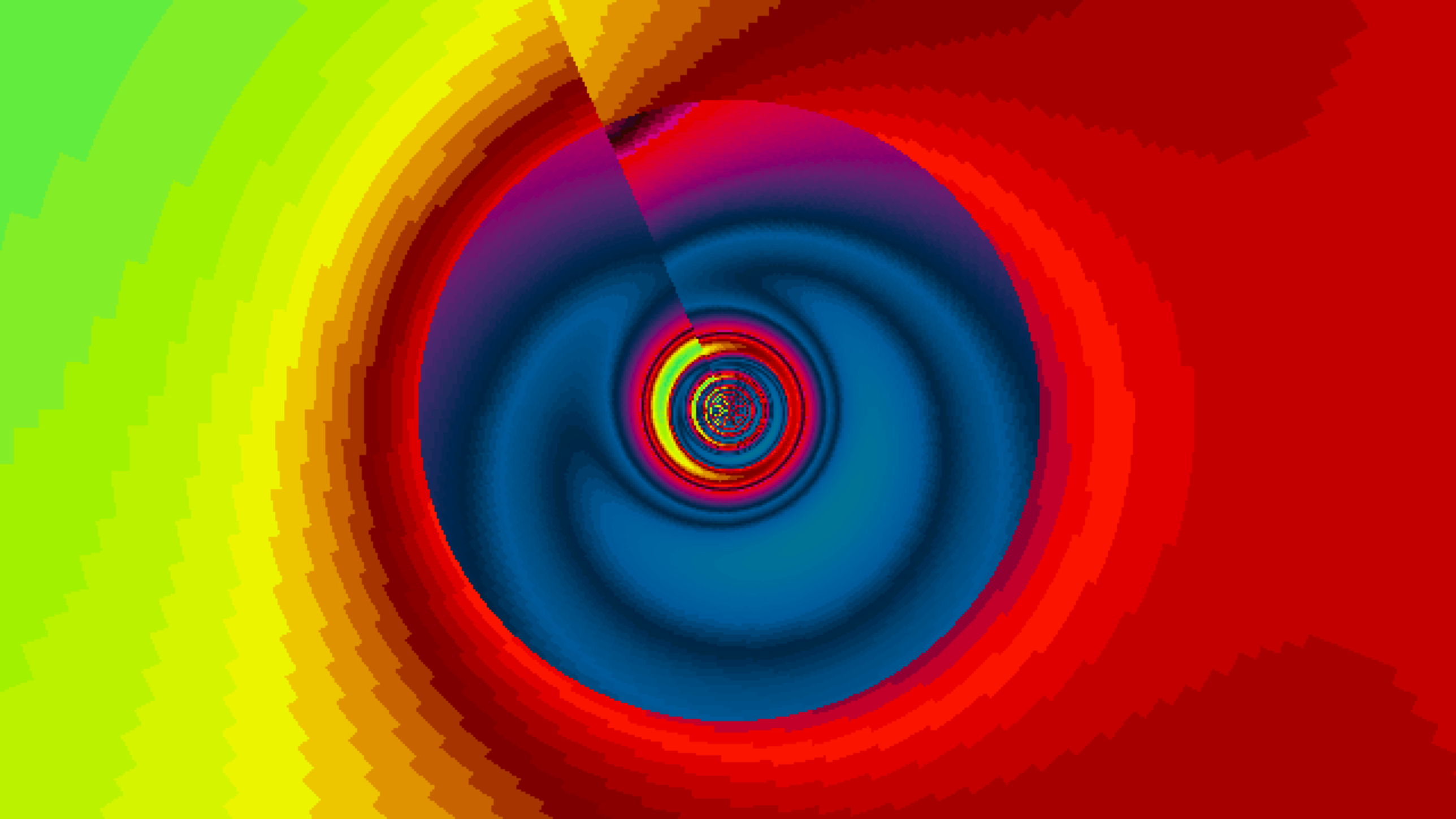
```
package main

import (
  "github.com/peterhellberg/gfx"
  "github.com/peterhellberg/plasma"
  "github.com/peterhellberg/plasma/palette"
)

func main() {
  w, h, scale, seed := 2560, 1440, 64.0, 1234

  gfx.SavePNG("plasma.png", plasma.New(w, h, scale).
    Image(w, h, seed, palette.MaterialDesign700),
  )
}
```



Tunnel

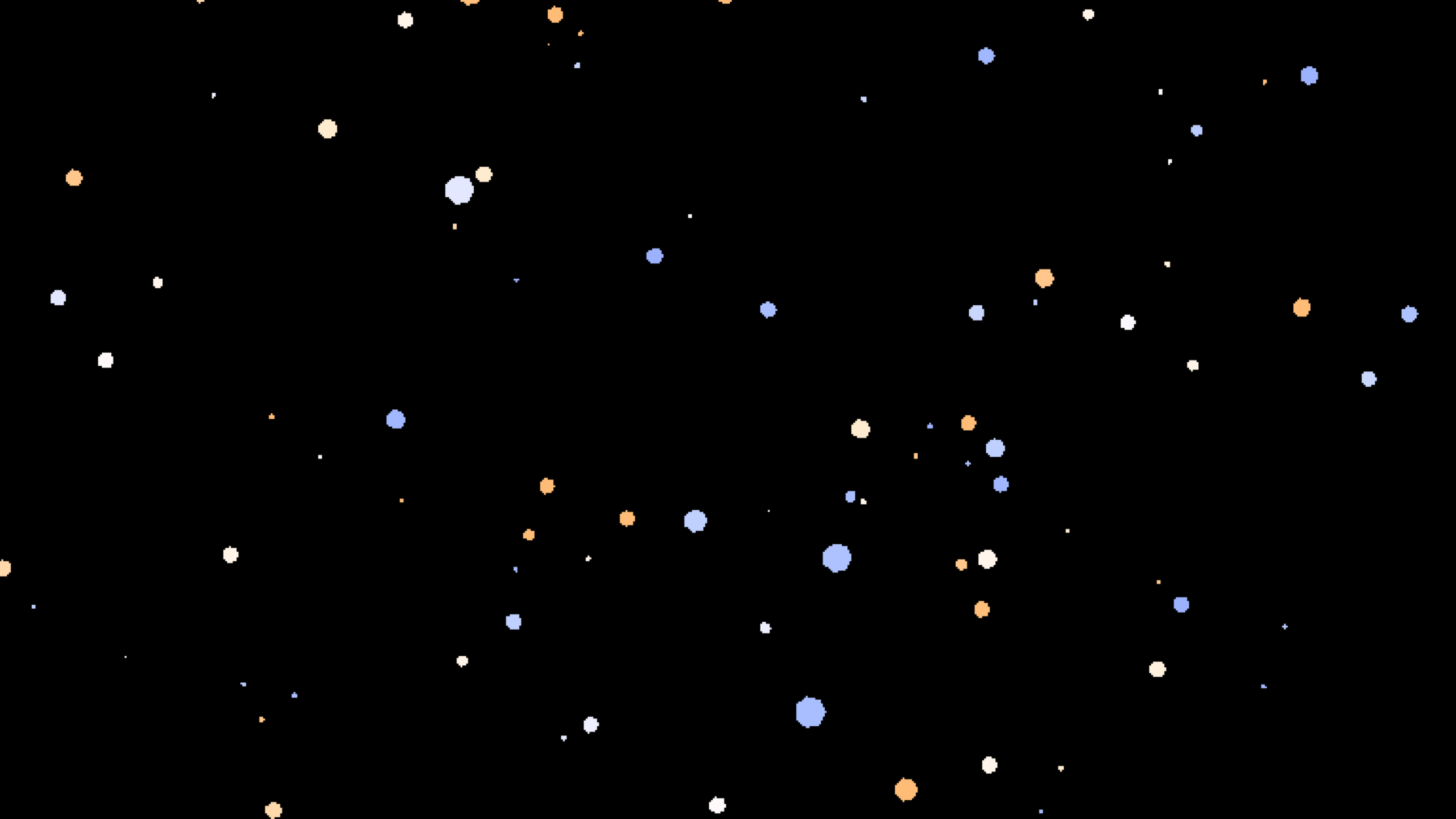
lodev.org/cgtutor/tunnel.html

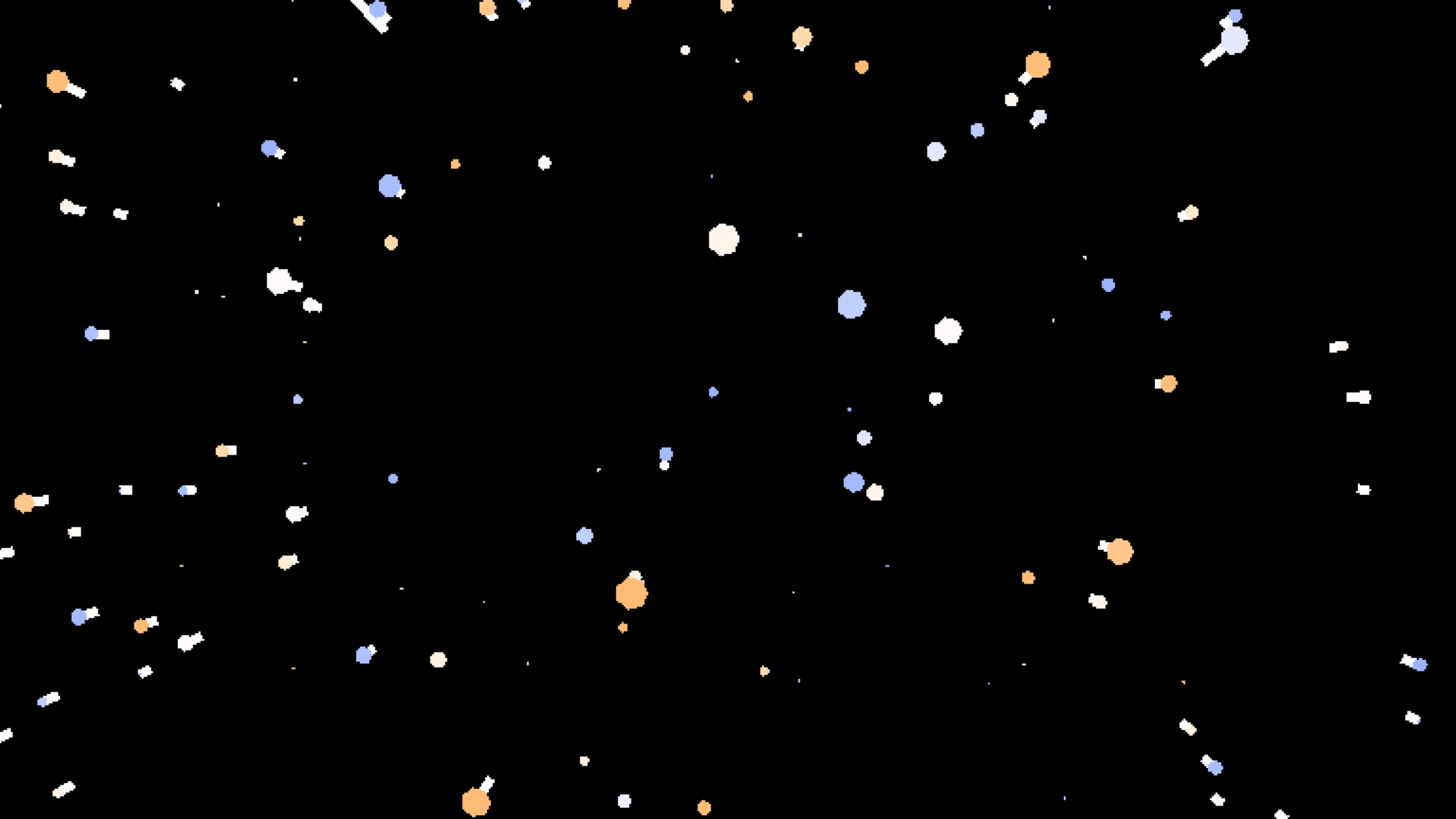


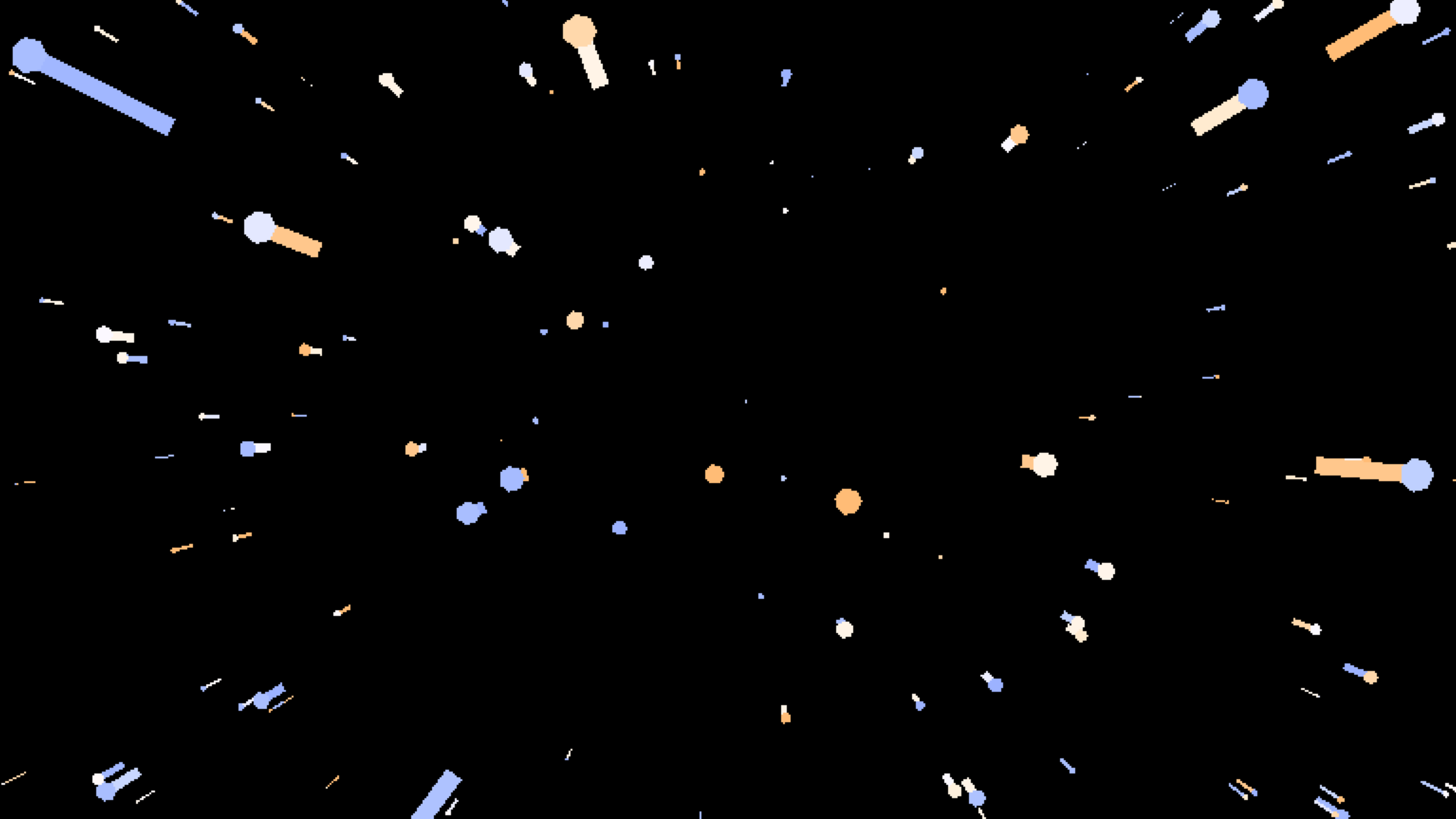


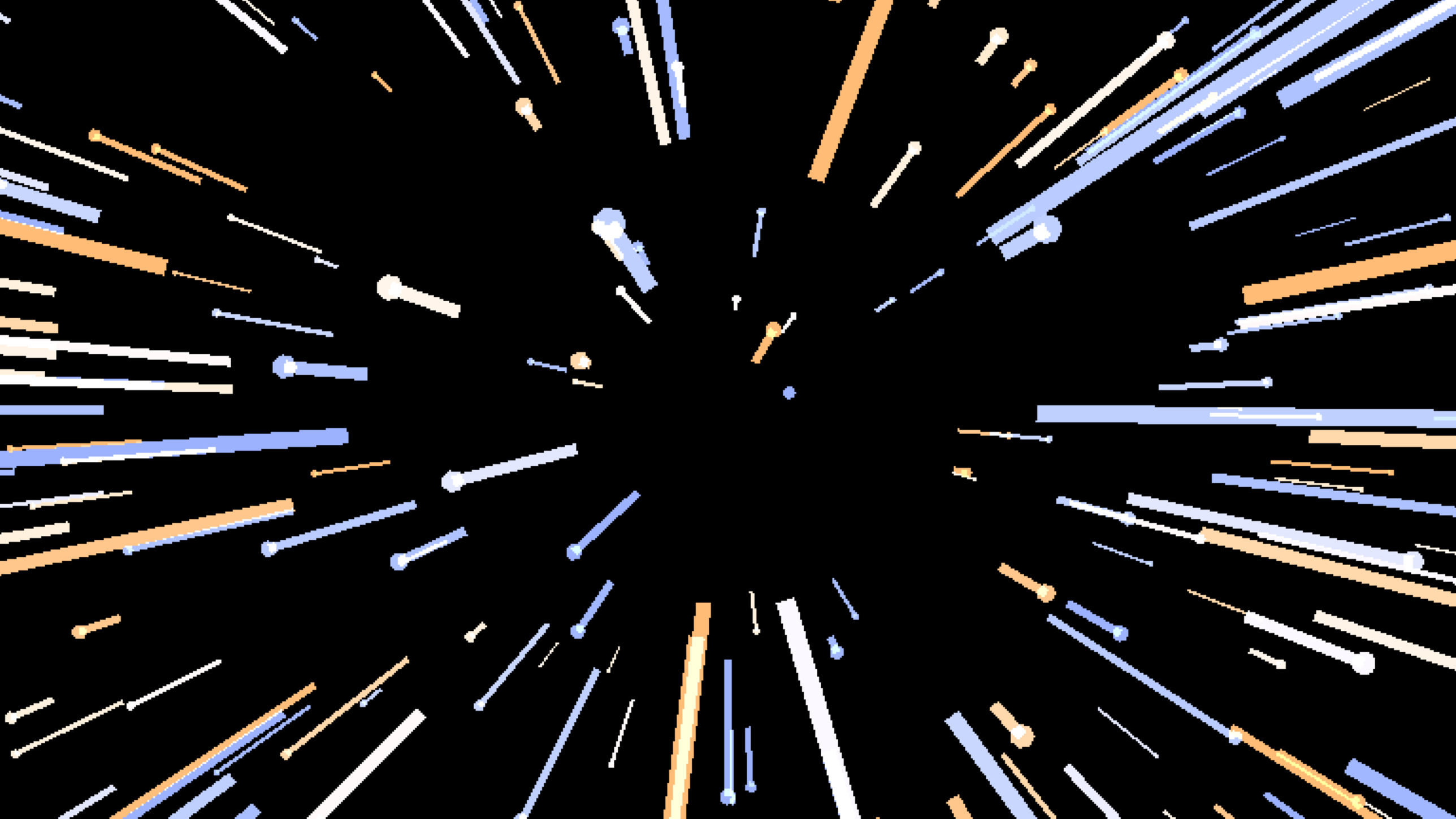
Starfield

vendian.org/mncharity/dir3/starcolor



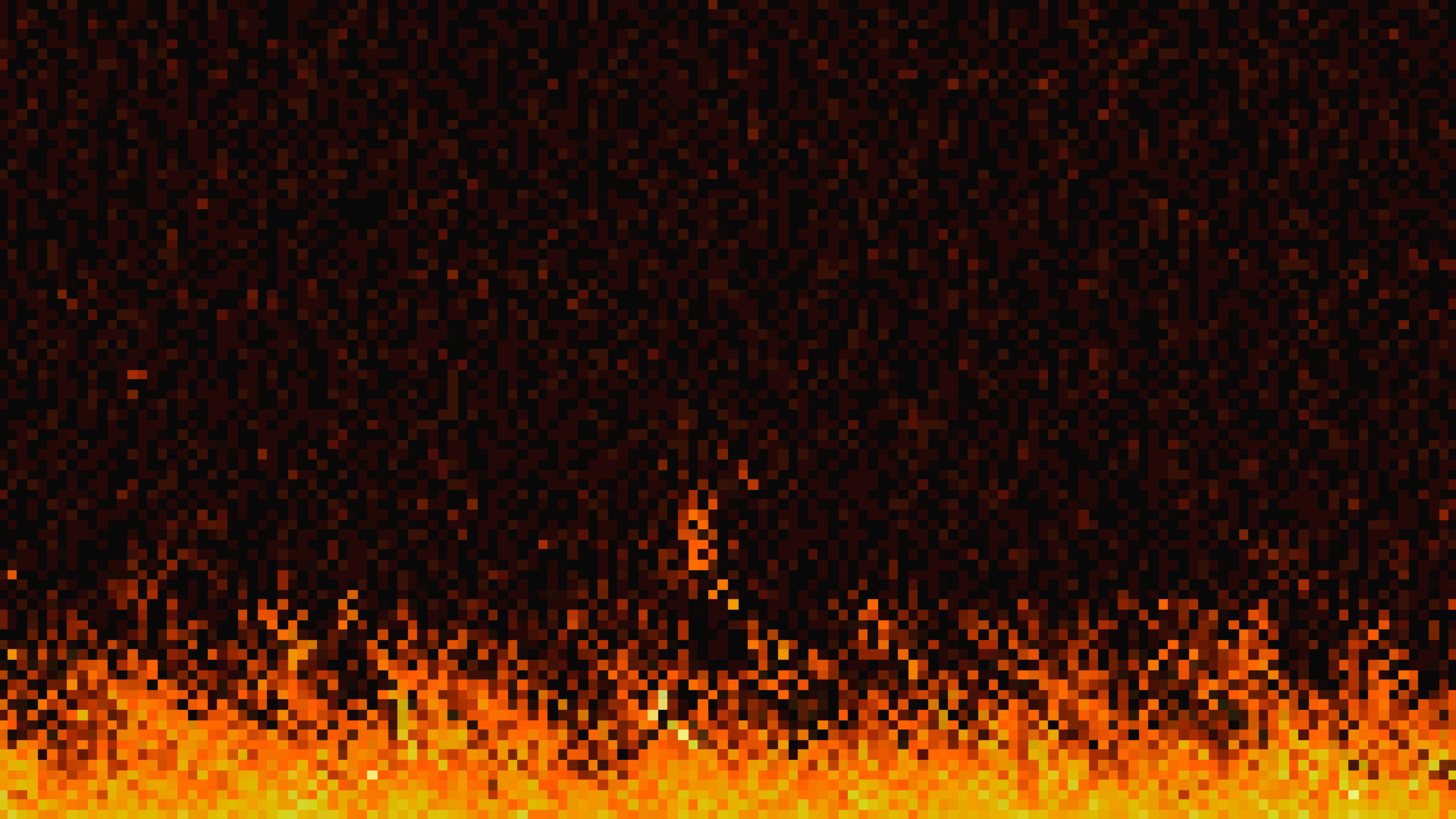




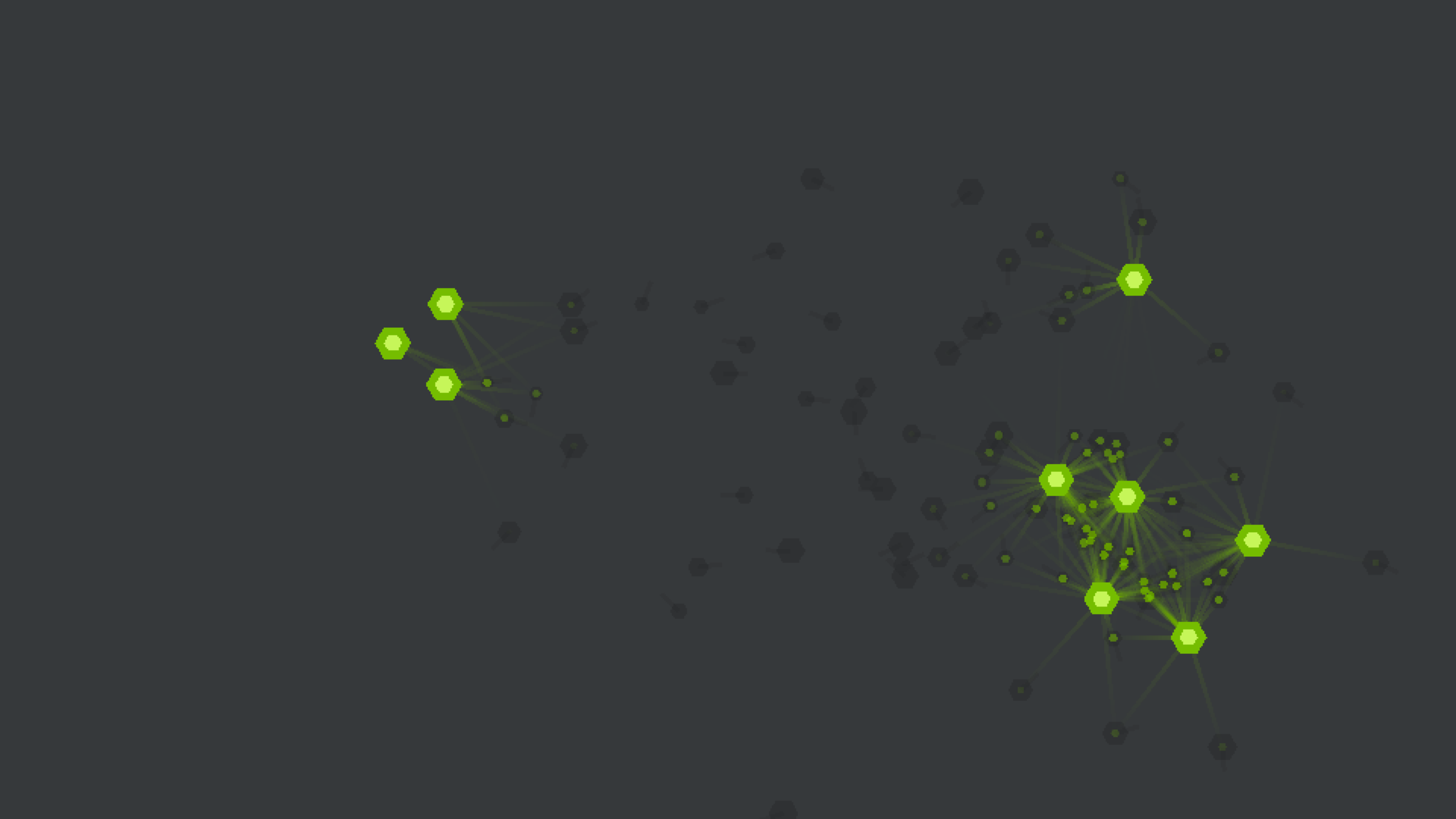


DOOM fire

fabiensanglard.net/doom_fire_psx

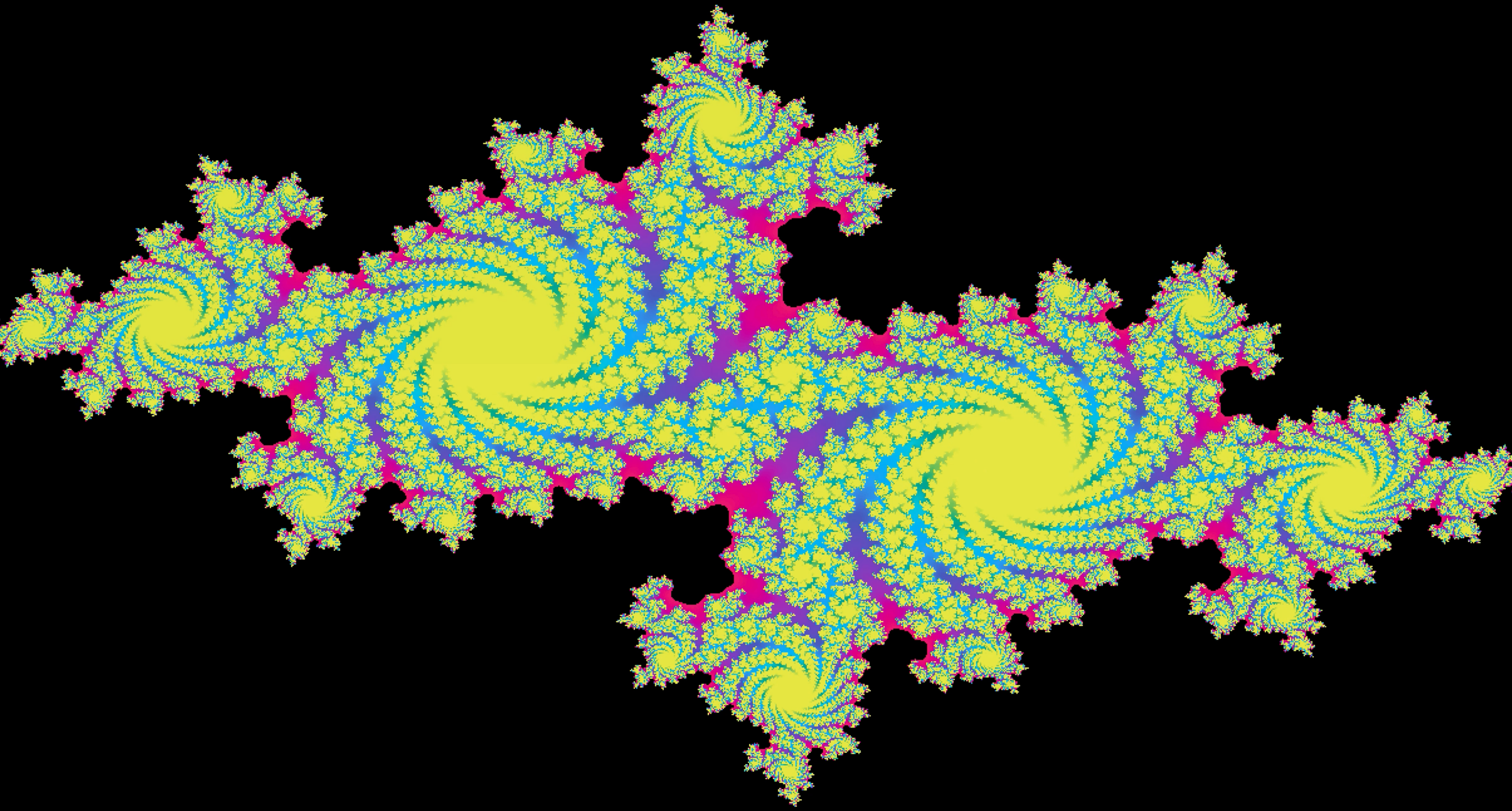


Particles



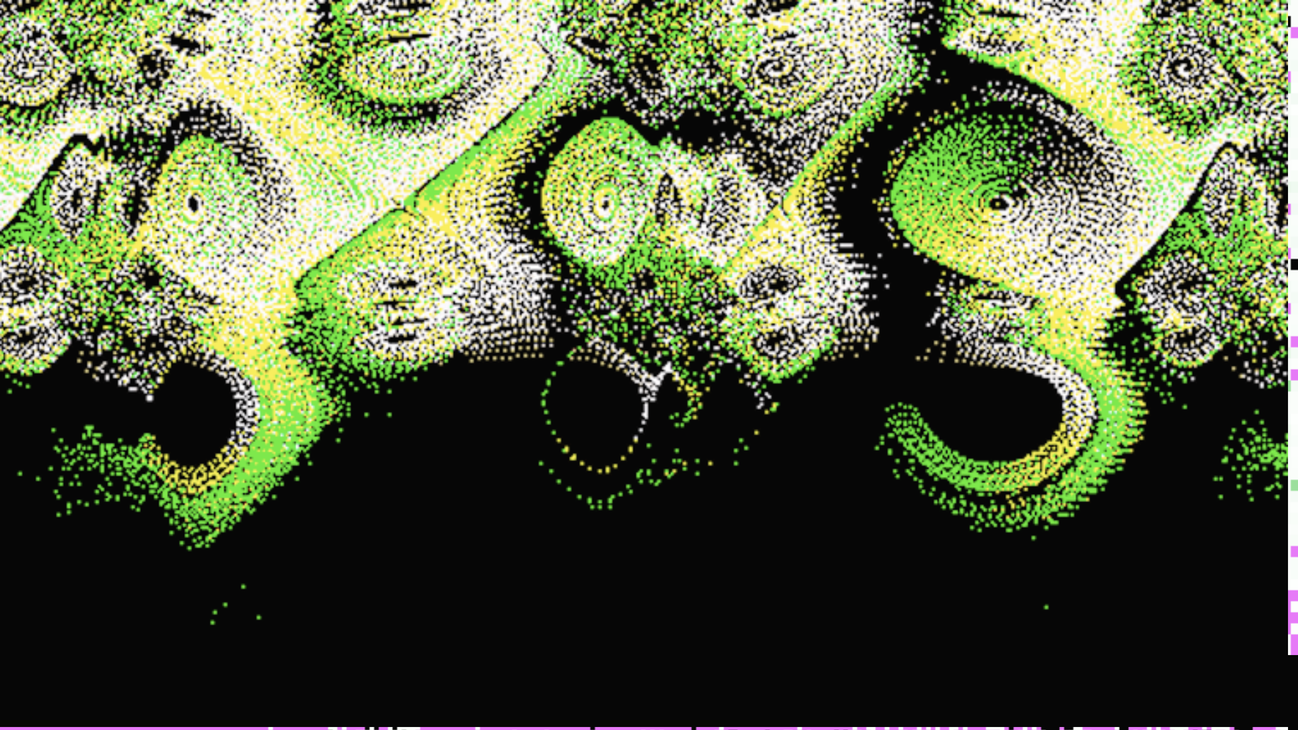
Fractals

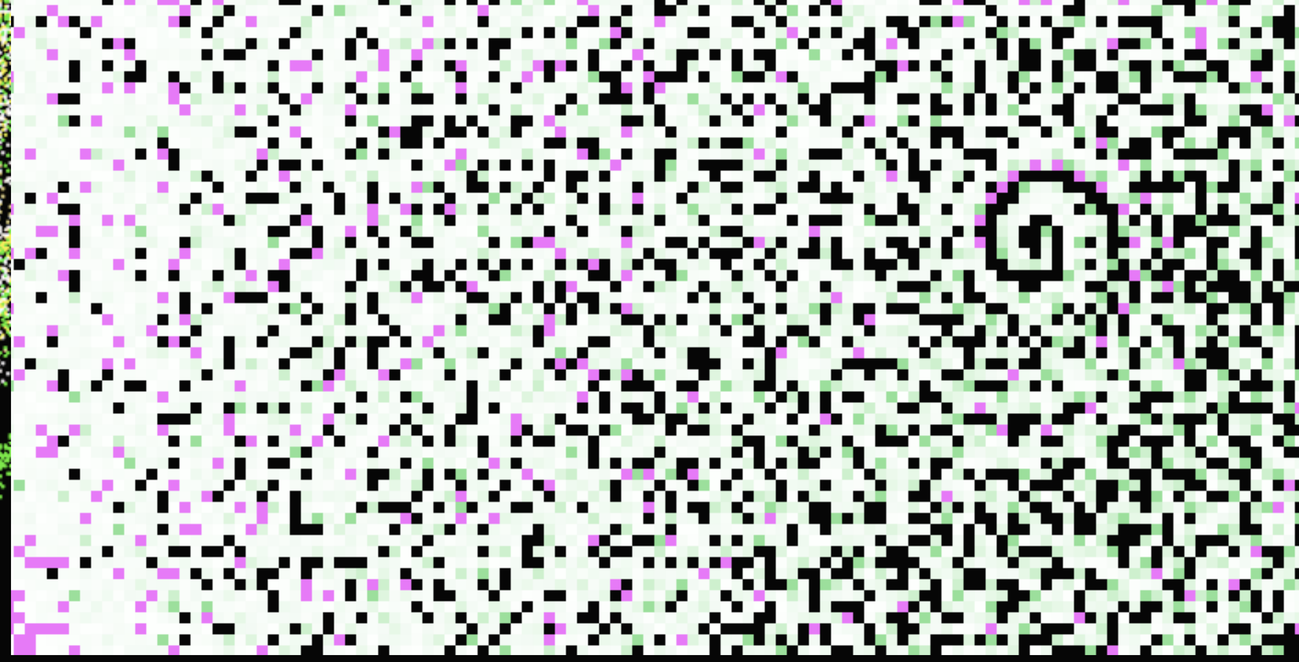
The Julia set

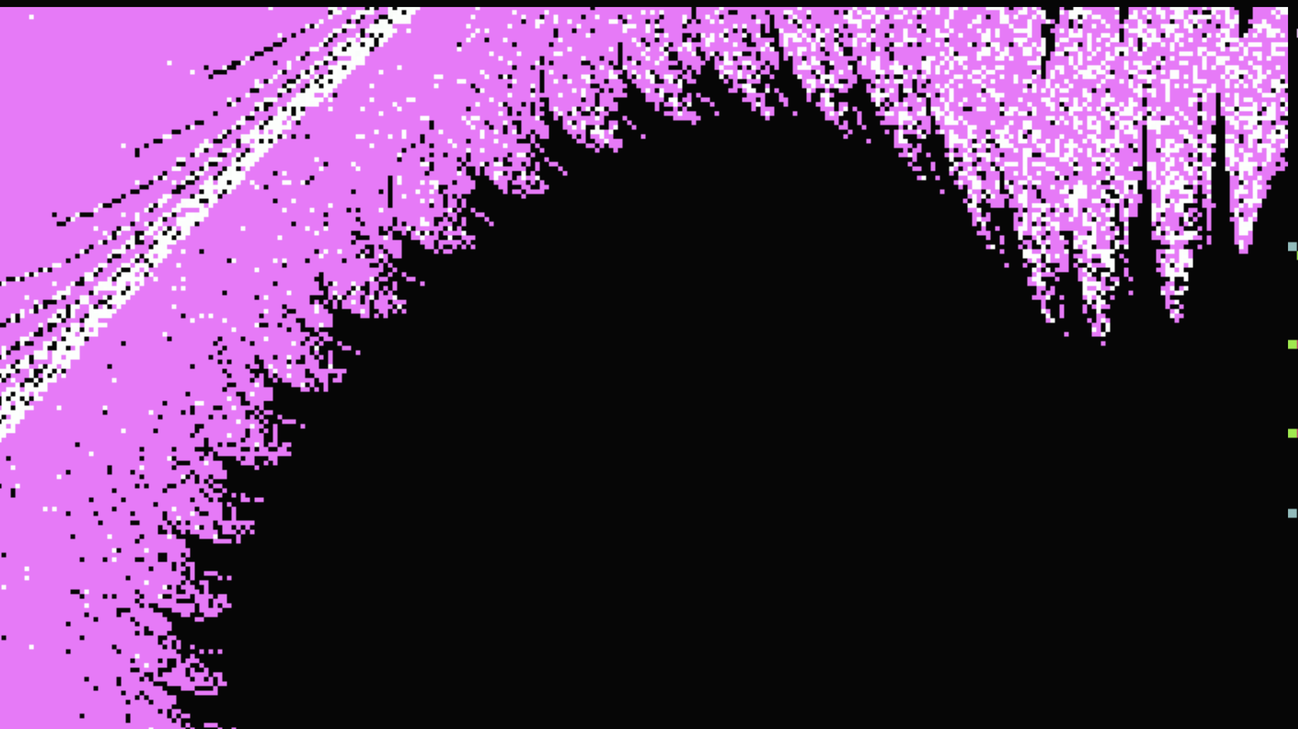


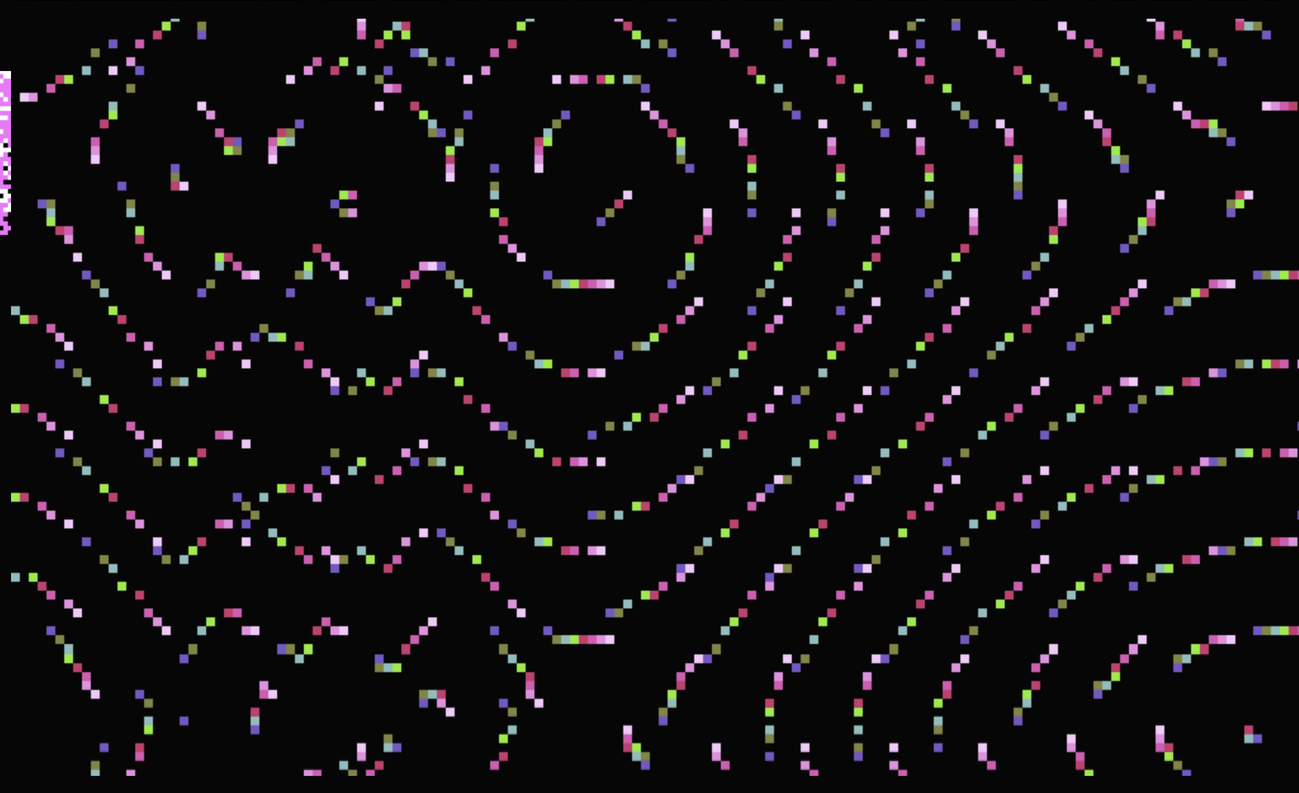
Popcorn fractals

paulbourke.net/fractals/popcorn

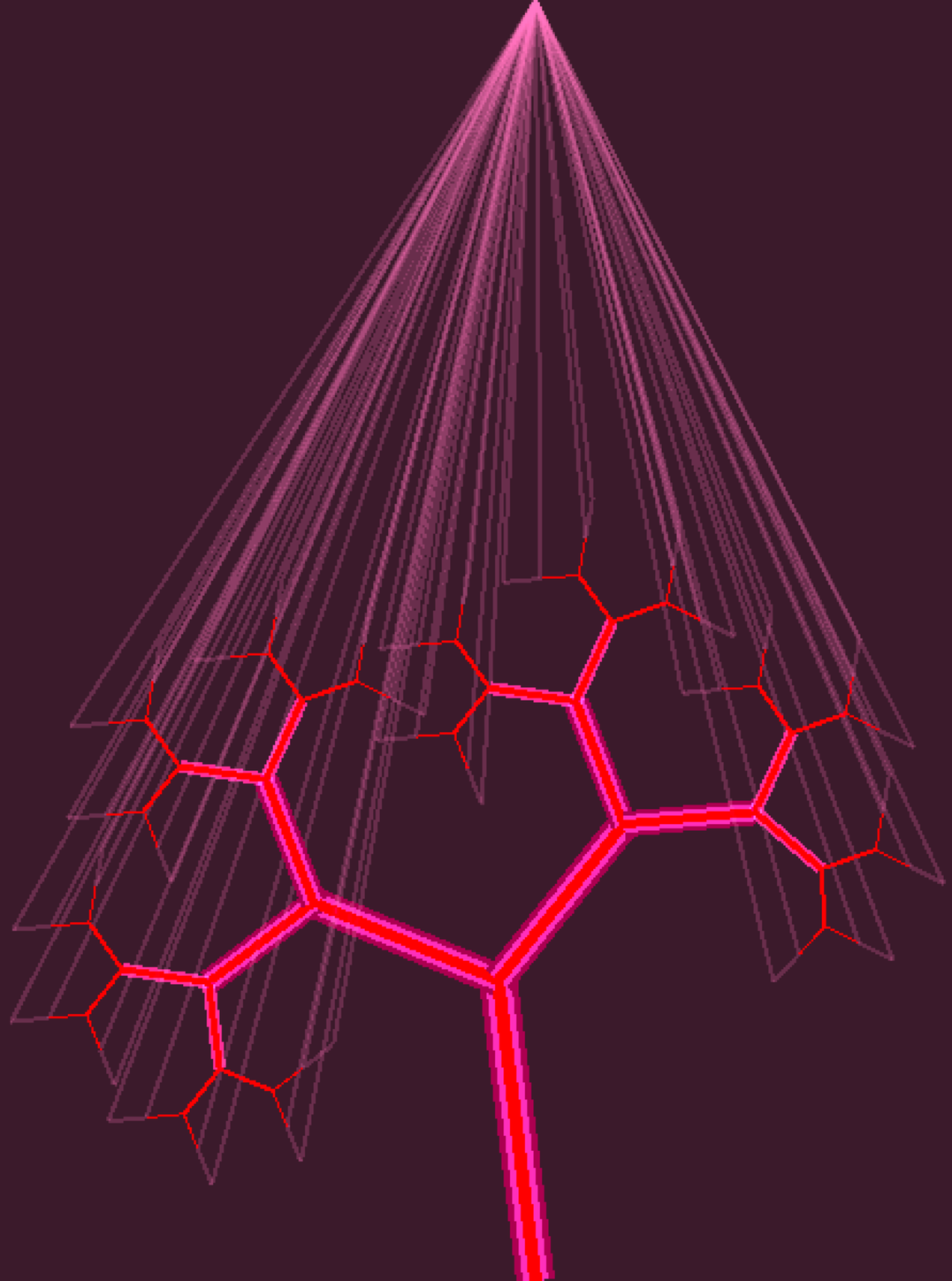


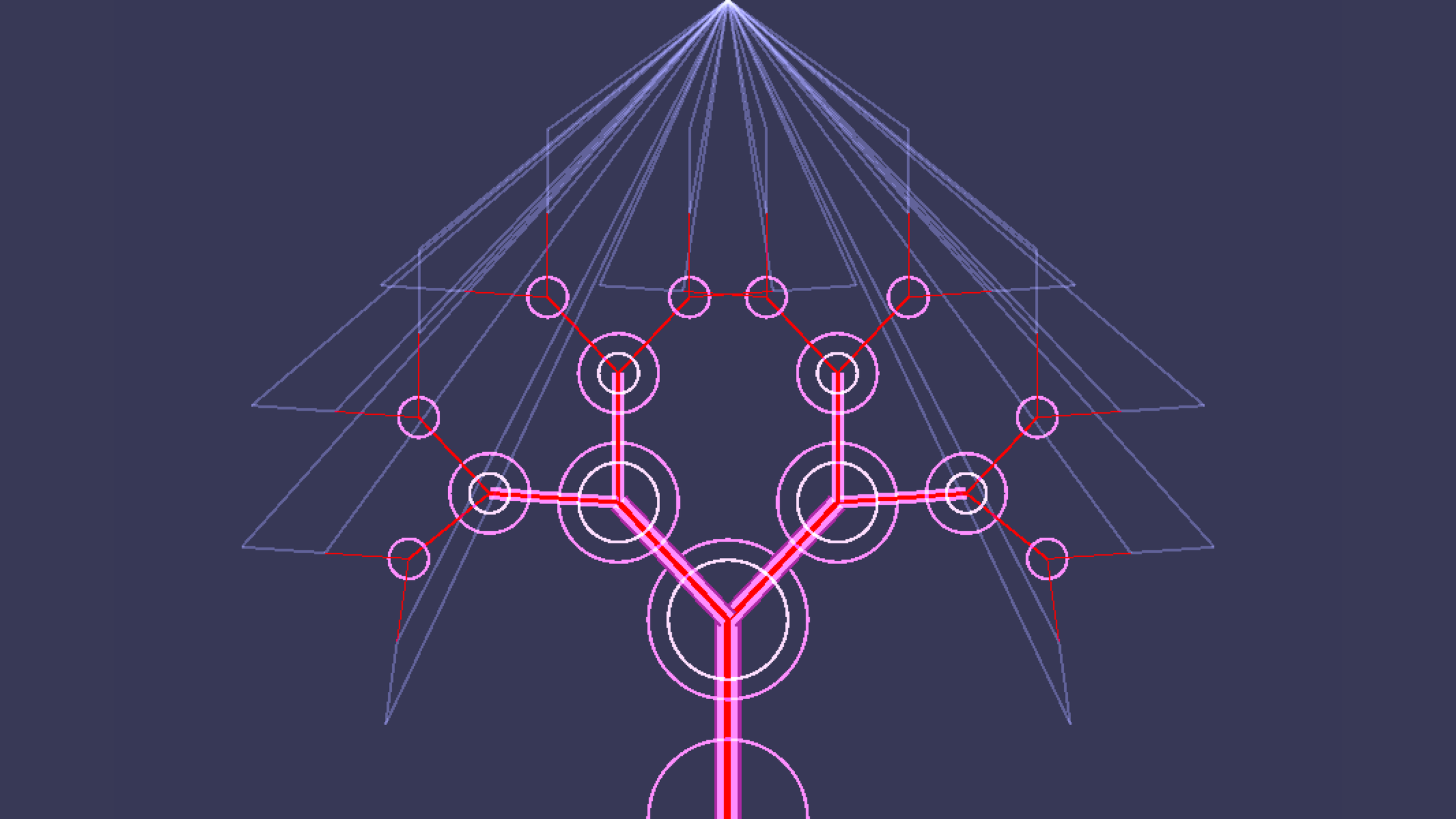


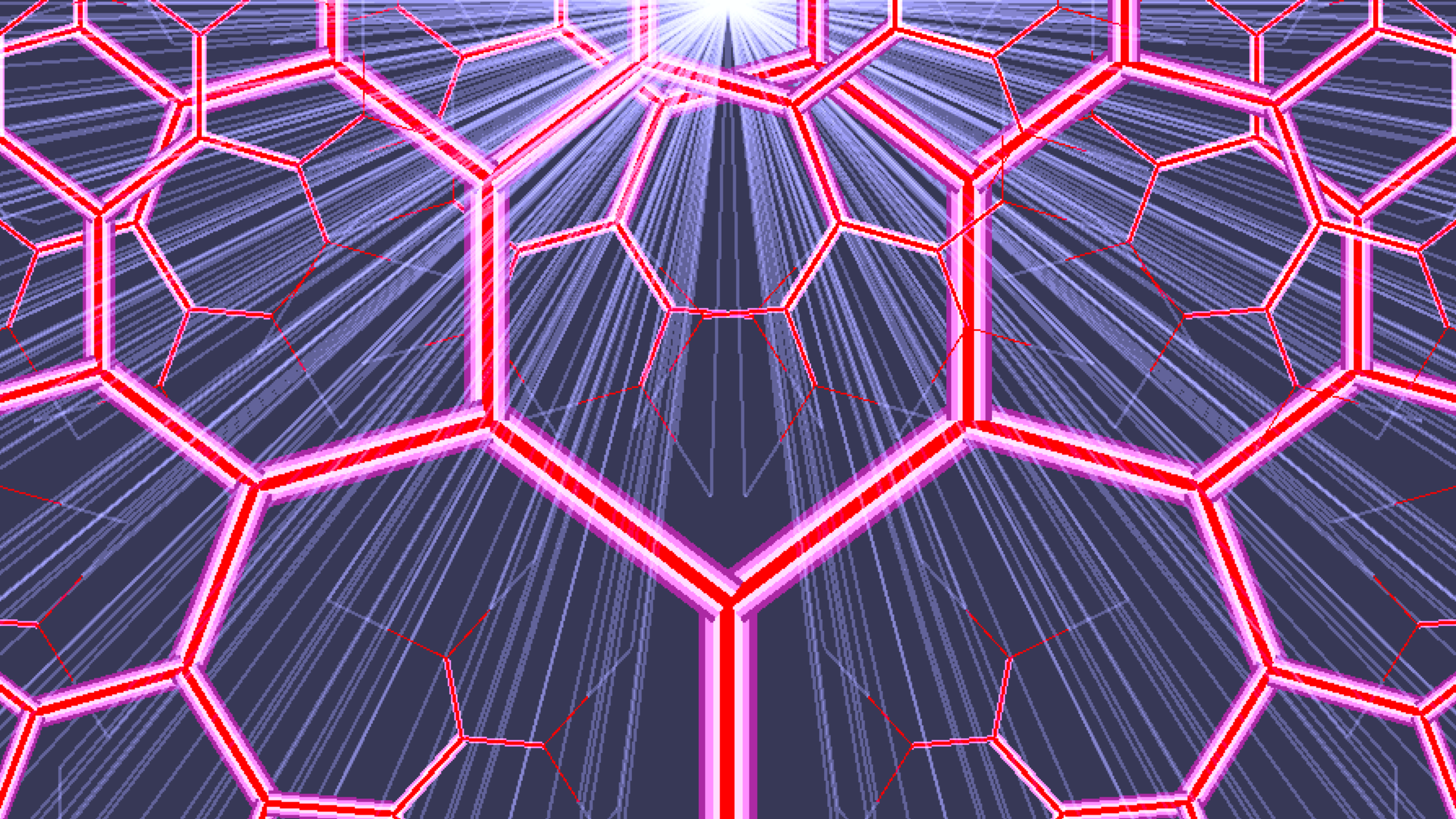


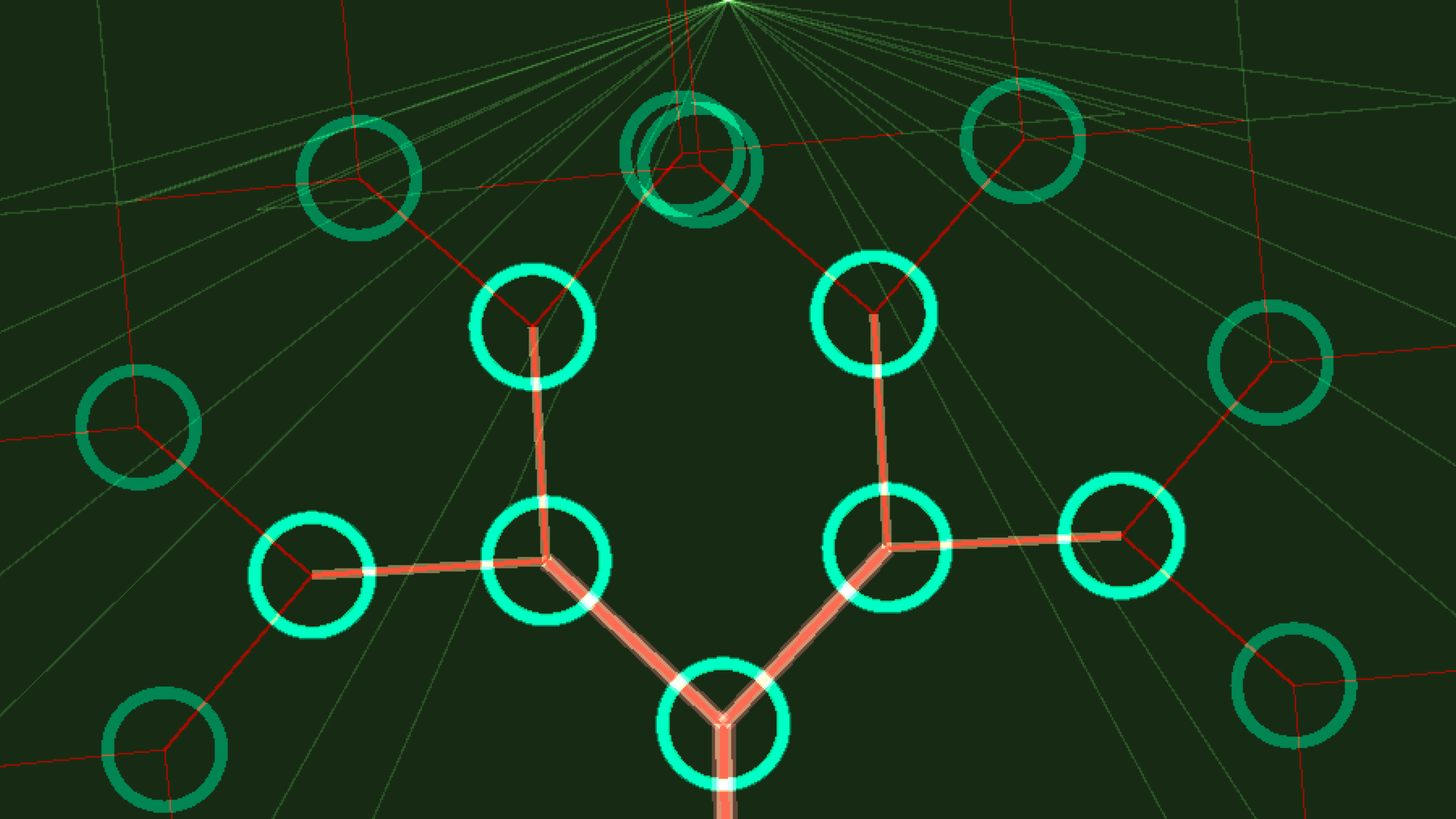


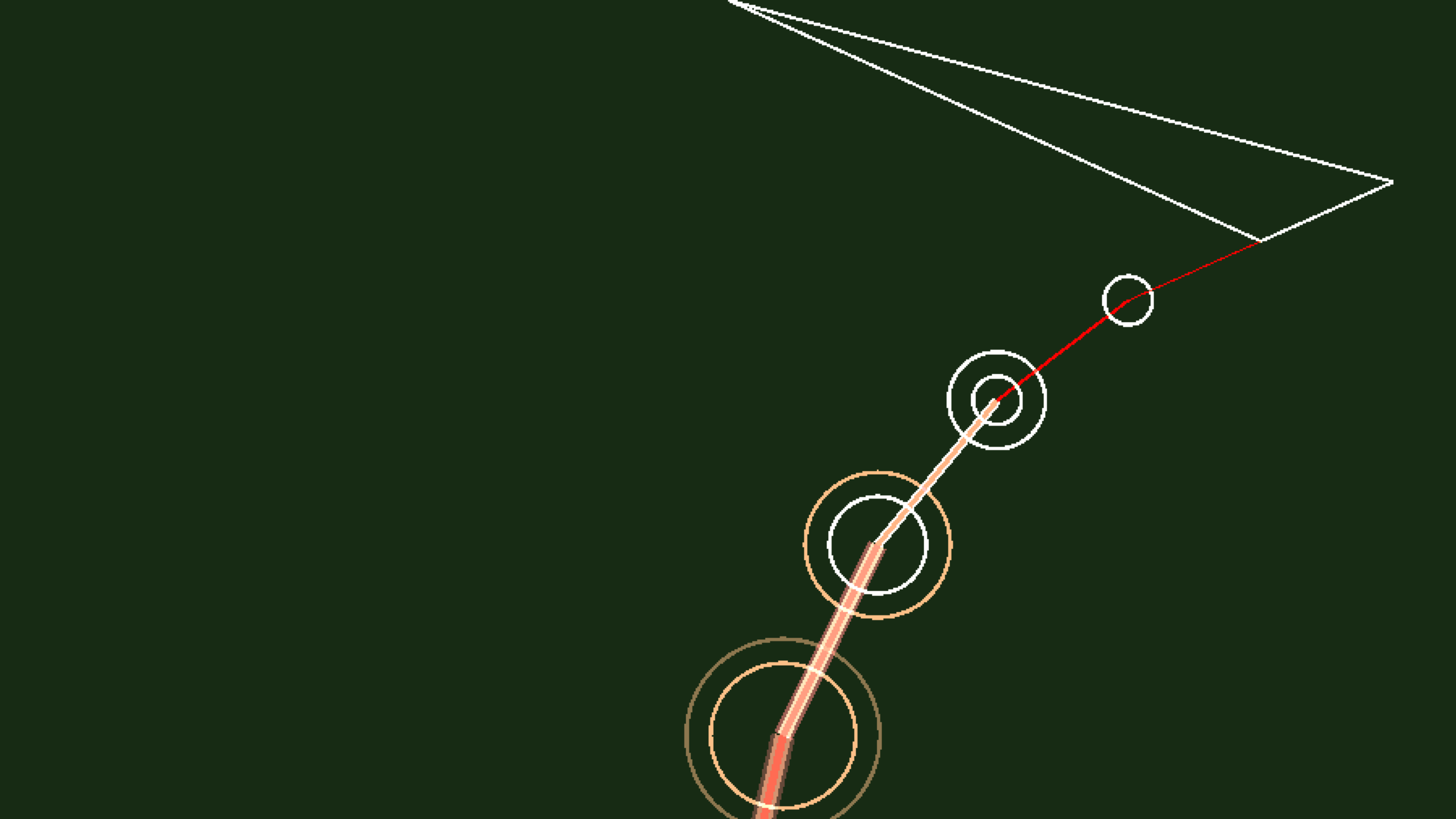
Procedural Tree





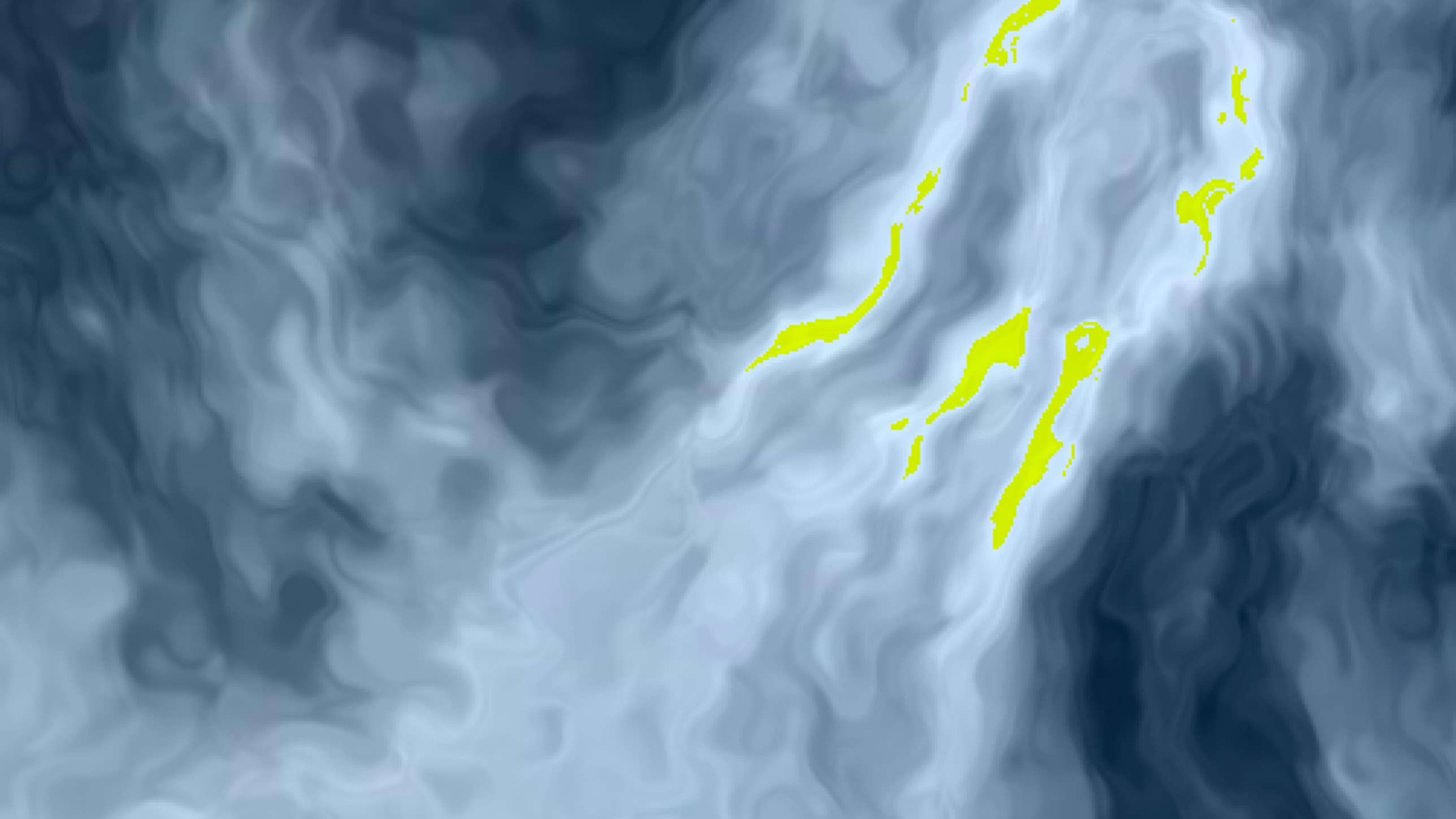


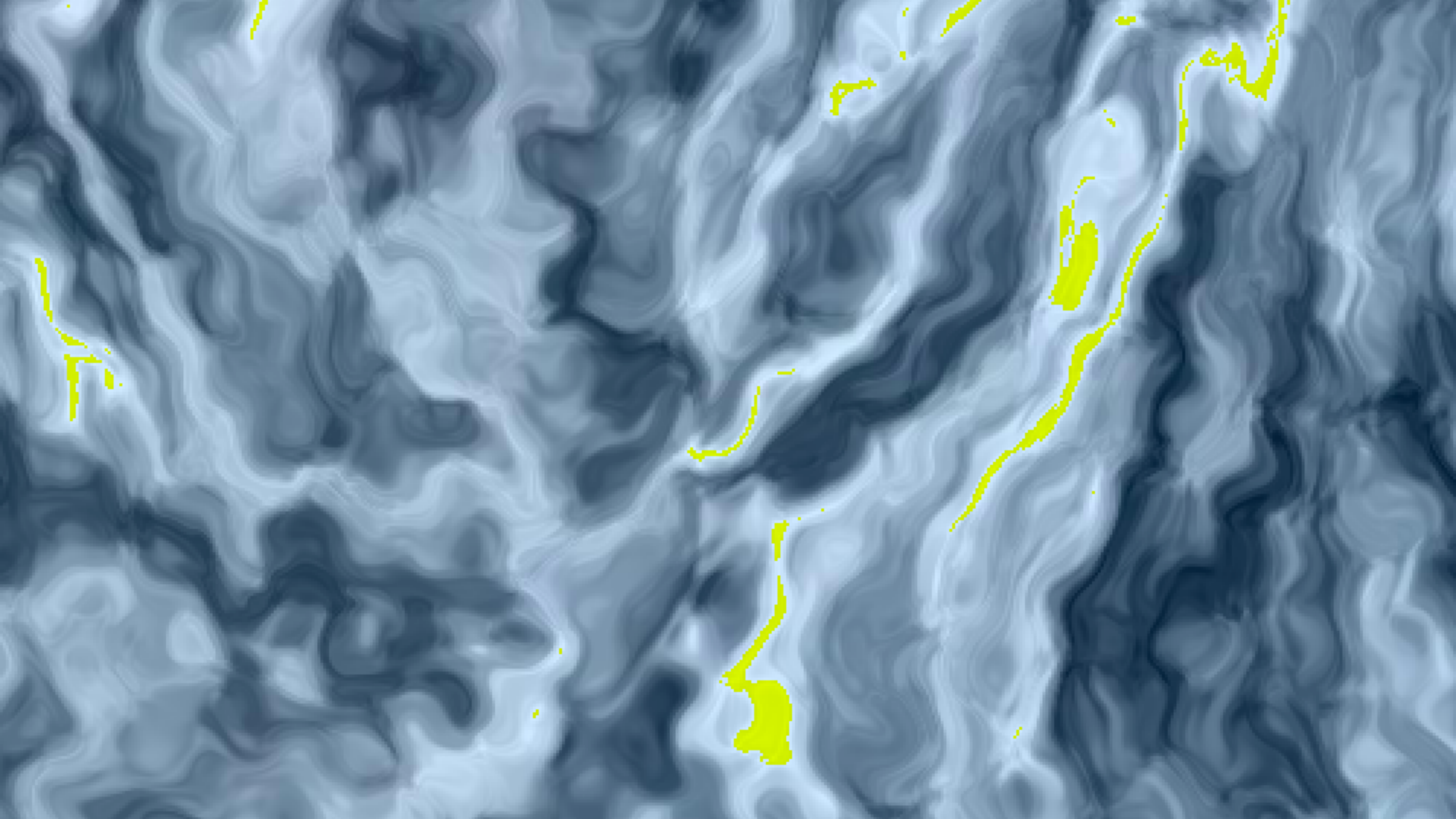


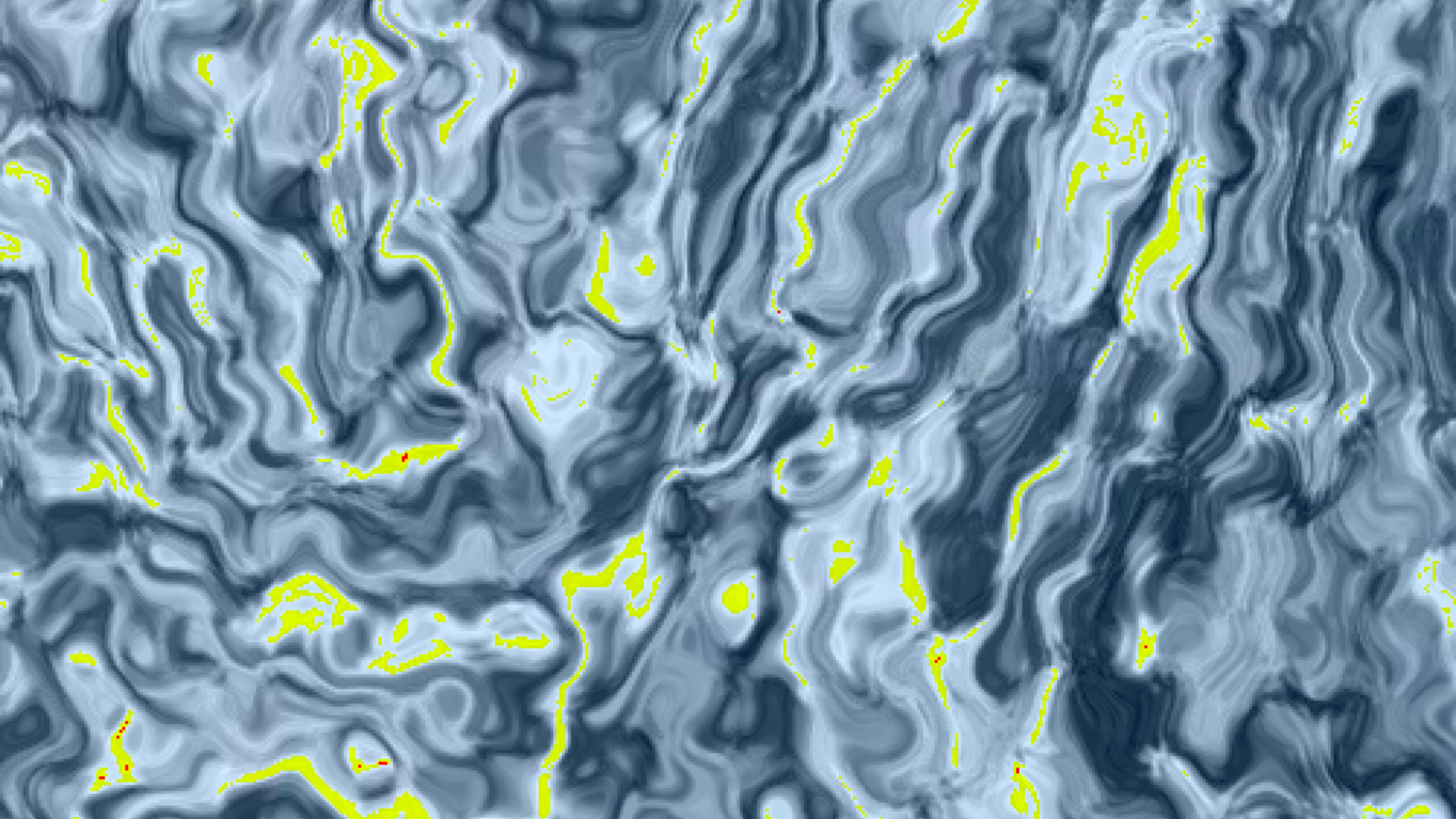


Domain warping

iquilezles.org/www/articles/warp/warp.htm

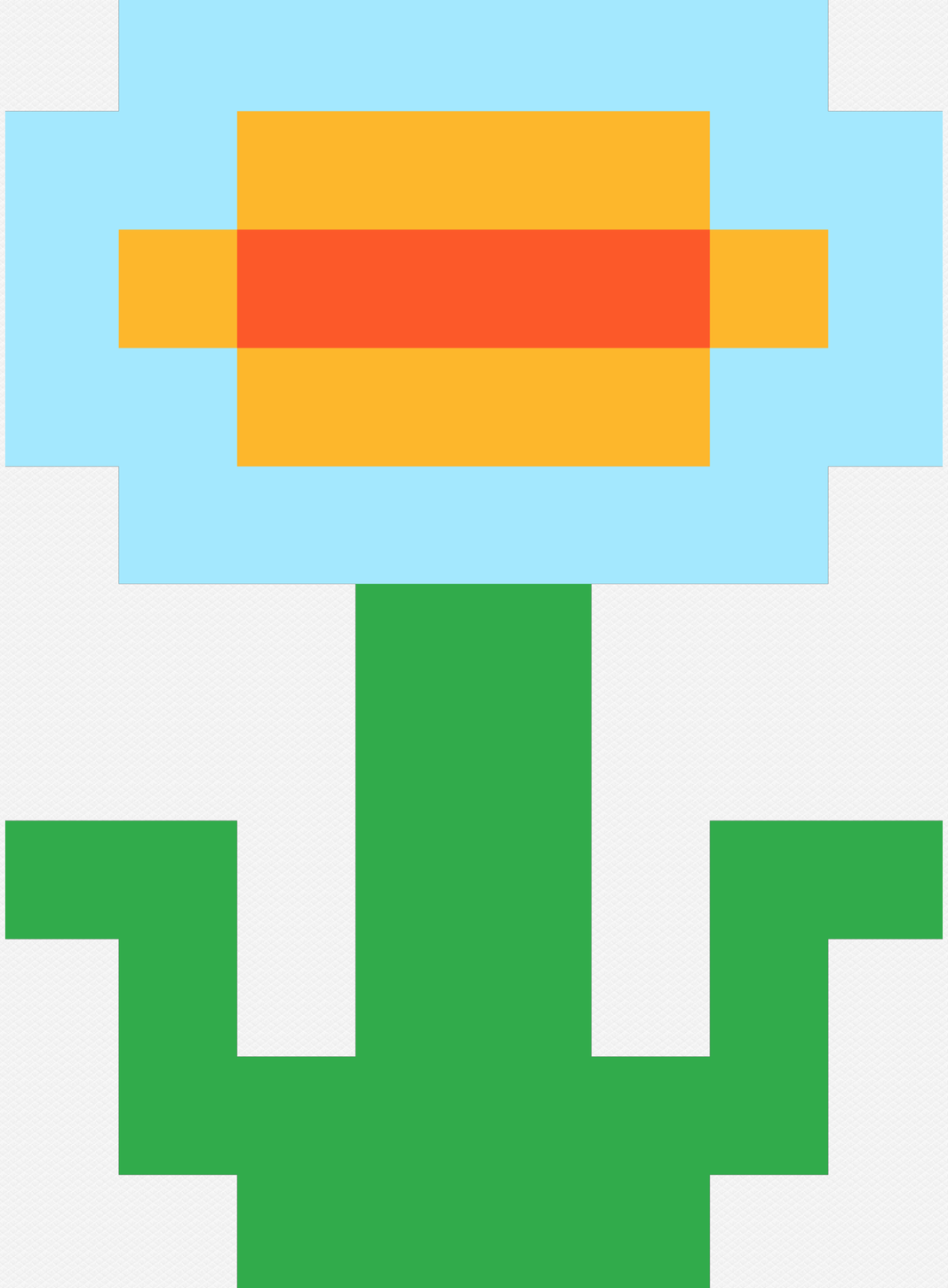




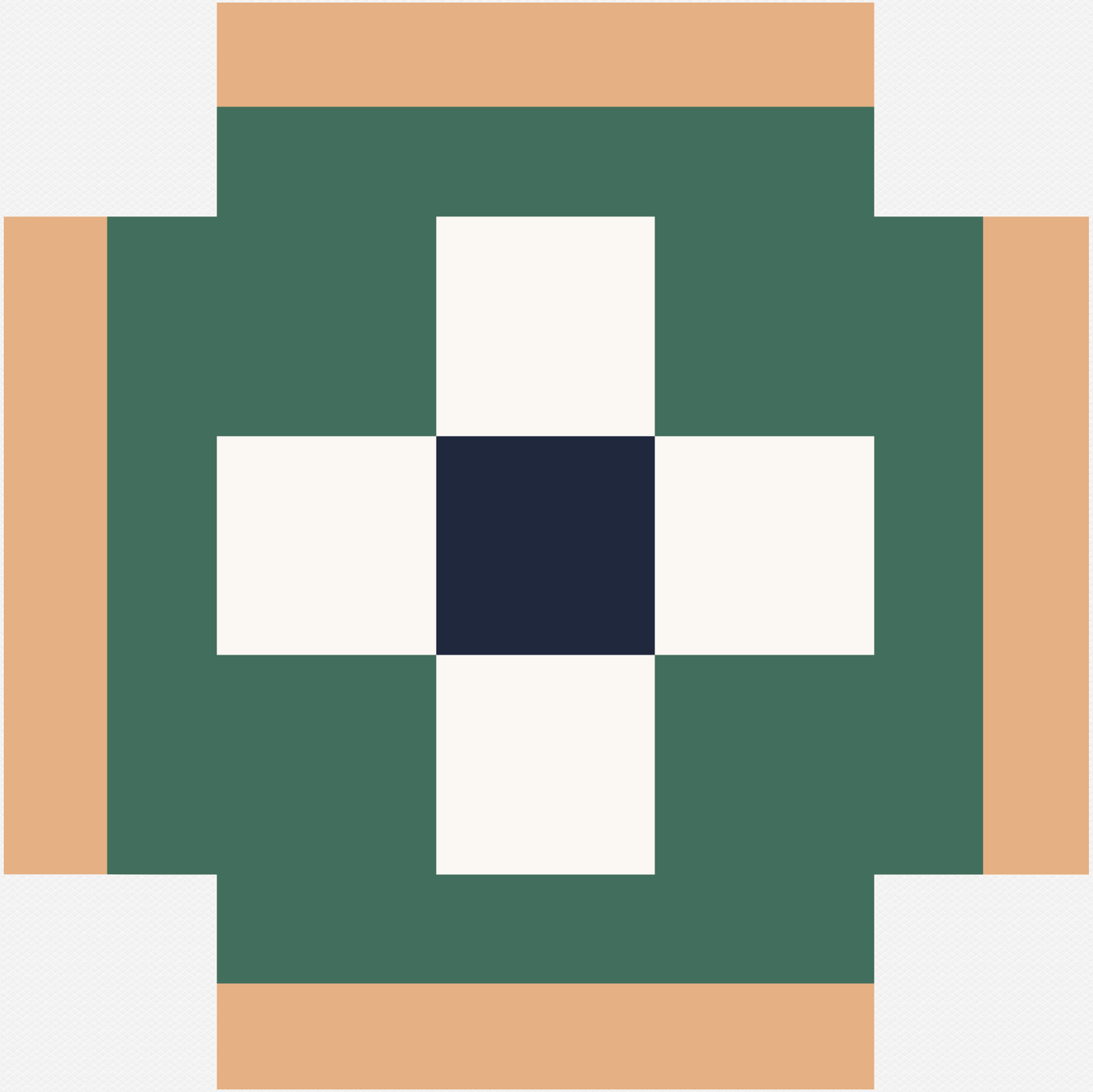


Tiles

```go
package main

import "github.com/peterhellberg/gfx"

func main() {
  gfx.SavePNG("fireflower.png", gfx.NewScaledImage(
    gfx.NewTile(palette, 8, []uint8{
      0, 1, 1, 1, 1, 1, 1, 0,
      1, 1, 2, 2, 2, 2, 1, 1,
      1, 2, 3, 3, 3, 3, 2, 1,
      1, 1, 2, 2, 2, 2, 1, 1,
      0, 1, 1, 1, 1, 1, 1, 0,
      0, 0, 0, 4, 4, 0, 0, 0,
      0, 0, 0, 4, 4, 0, 0, 0,
      4, 4, 0, 4, 4, 0, 4, 4,
      0, 4, 0, 4, 4, 0, 4, 0,
      0, 4, 4, 4, 4, 4, 4, 0,
      0, 0, 4, 4, 4, 4, 0, 0,
    }), 128))
}

var palette = gfx.Palette{{},
  {0xA4, 0xE8, 0xFE, 0xFF},
  {0xFD, 0xB7, 0x2C, 0xFF},
  {0xFC, 0x59, 0x29, 0xFF},
  {0x31, 0xAB, 0x4B, 0xFF},
}
```



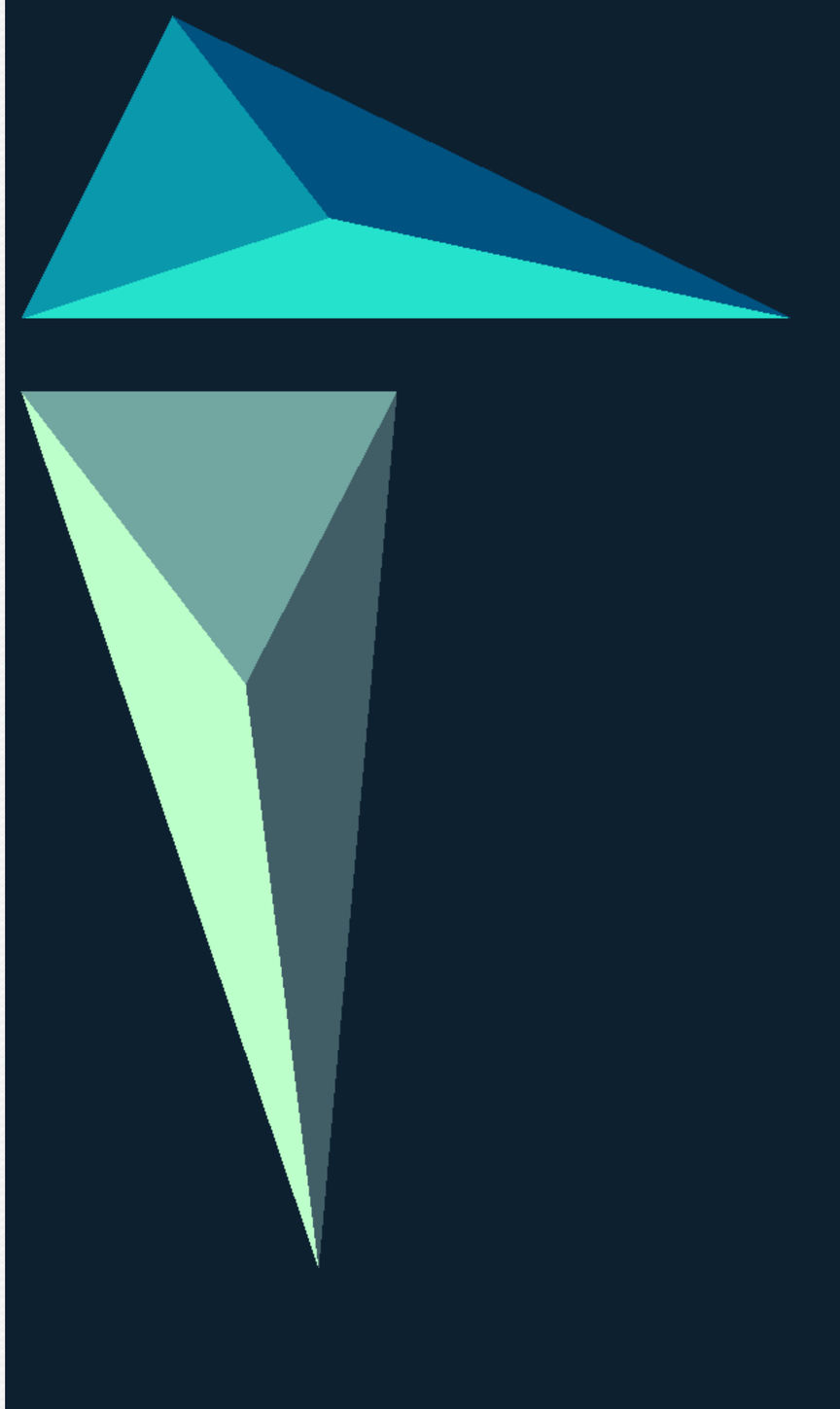
```go
package main

import "github.com/peterhellberg/gfx"

func main() {
  gfx.SavePNG("layer.png", gfx.NewScaledImage(
    gfx.NewLayer(tileset, 5, gfx.LayerData{
      4, 7, 7, 7, 4,
      5, 2, 0, 2, 6,
      5, 0, 3, 0, 6,
      5, 2, 0, 2, 6,
      4, 8, 8, 8, 4,
    }), 128),
  )
}

var tileset = gfx.NewTileset(
 gfx.PaletteEN4, gfx.Pt(2, 2),
 gfx.TilesetData{
  {0, 0, 0, 0}, {1, 1, 1, 1}, {2, 2, 2, 2},
  {3, 3, 3, 3}, {9, 9, 9, 9}, {1, 2, 1, 2},
  {2, 1, 2, 1}, {1, 1, 2, 2}, {2, 2, 1, 1},
})
```



Triangles

Triangles

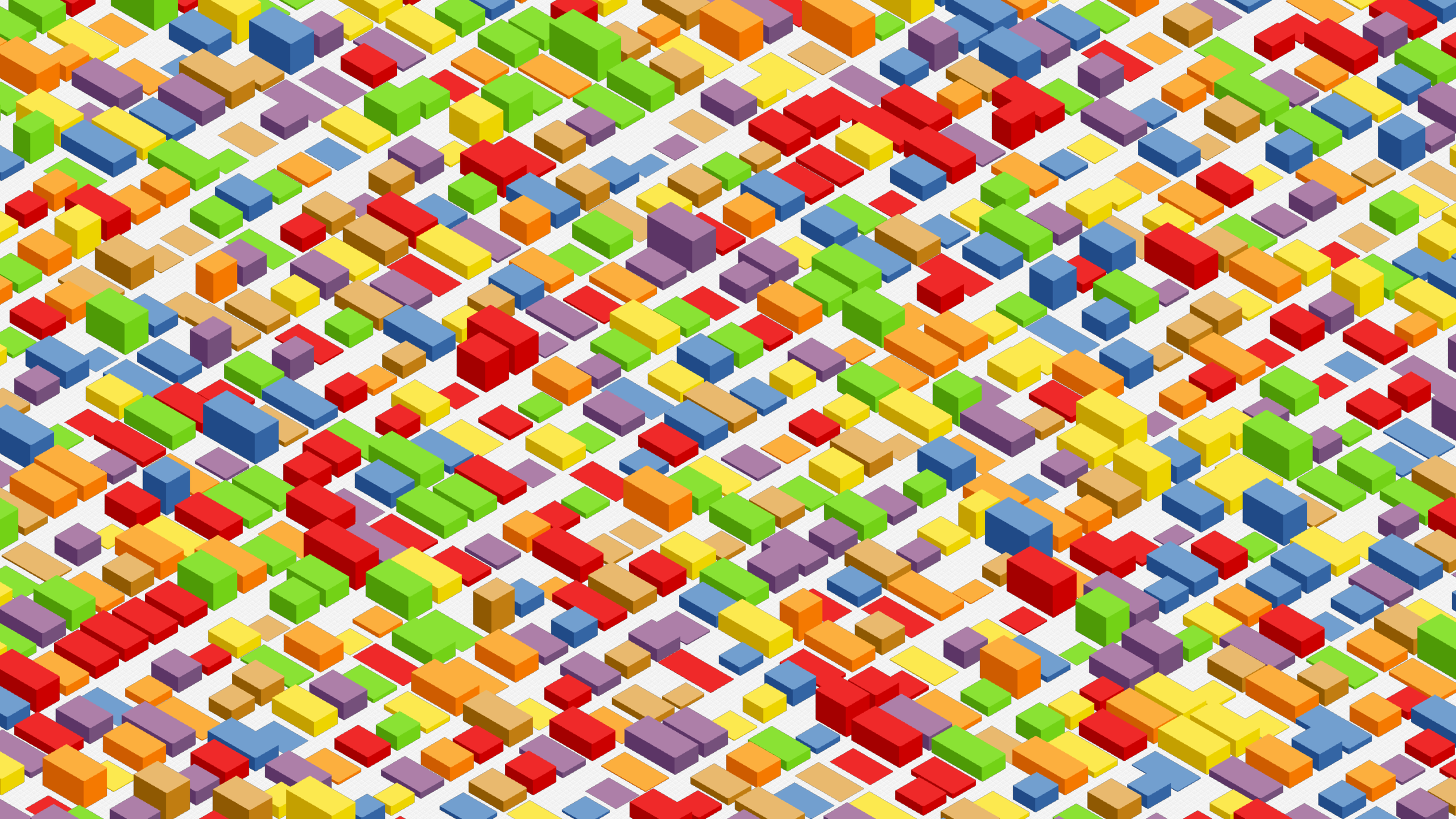
```
package main

import "github.com/peterhellberg/gfx"

var p = gfx.PaletteFamicube

func vx(x, y float64, n int) gfx.Vertex {
  return gfx.Vertex{Position: gfx.V(x, y), Color: p.Color(n)}
}

func main() {
  m := gfx.NewPaletted(640, 1080, p, p.Color(57))

  gfx.NewDrawTarget(m).MakeTriangles(&gfx.TrianglesData{
    vx(128, 12, 51), vx(12, 244, 52), vx(604, 244, 53),
    vx(12, 300, 54), vx(300, 300, 55), vx(240, 972, 56),
  }).Draw()

  gfx.SavePNG("triangles.png", m)
}
```

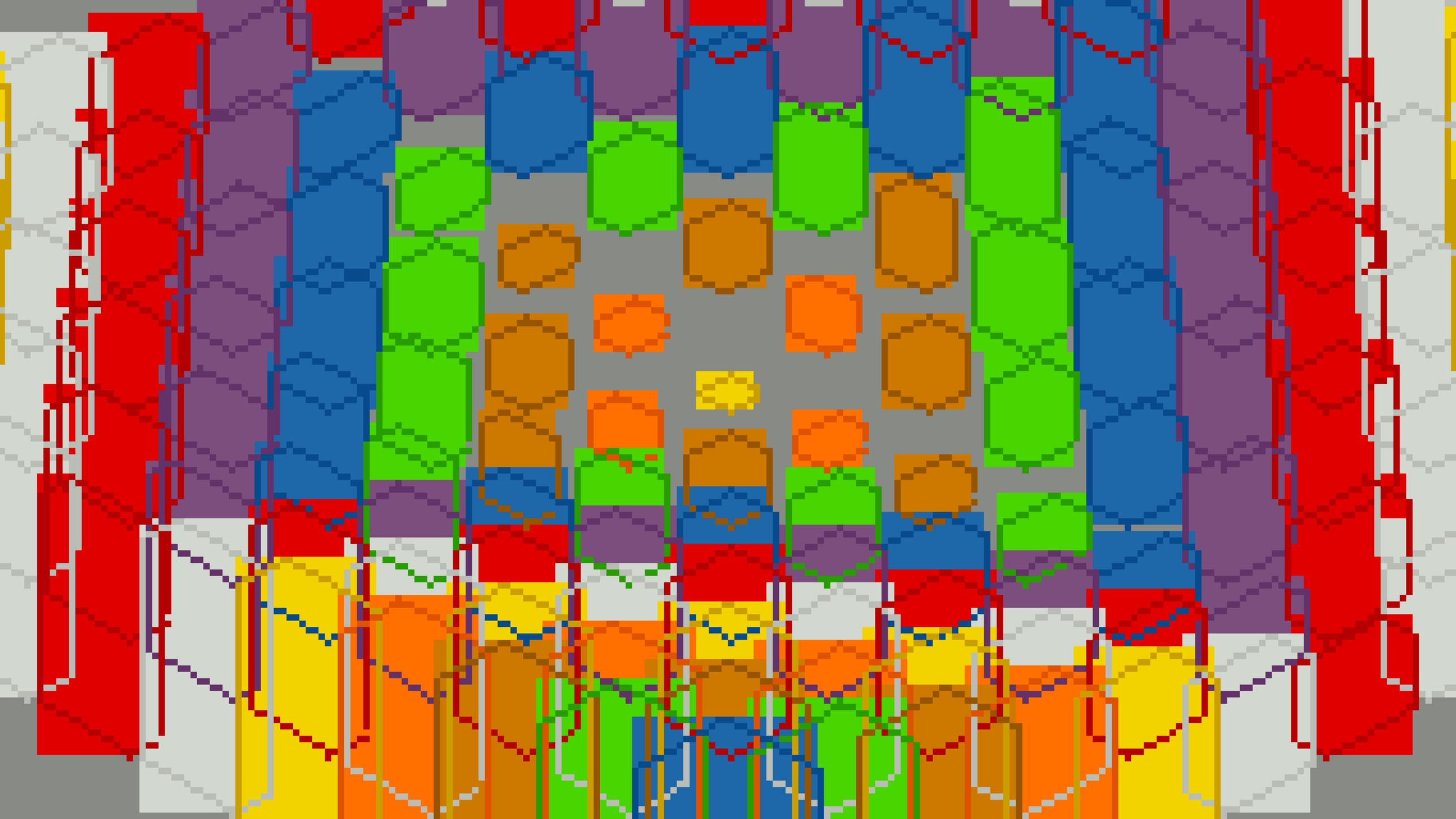


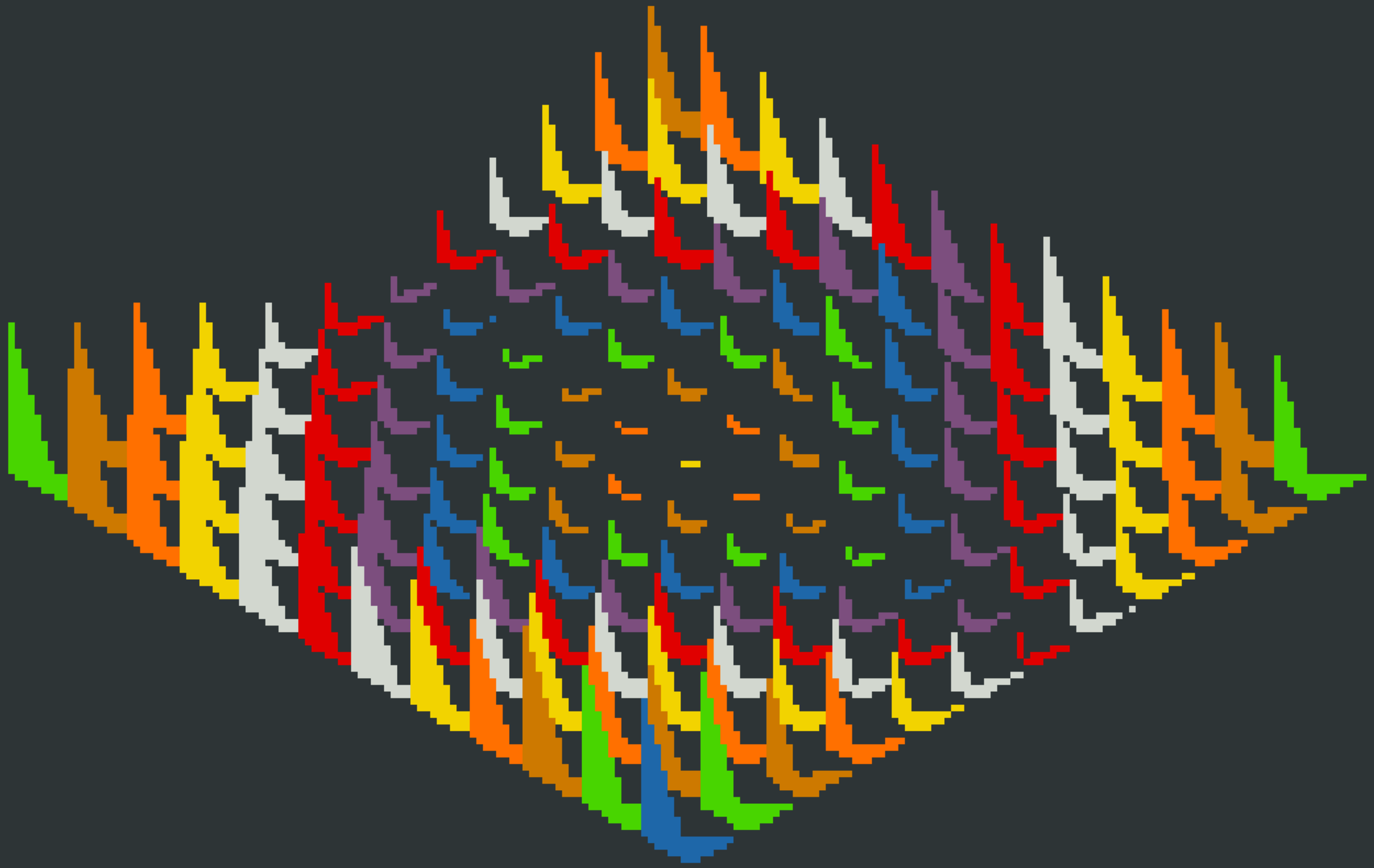
Isometric blocks

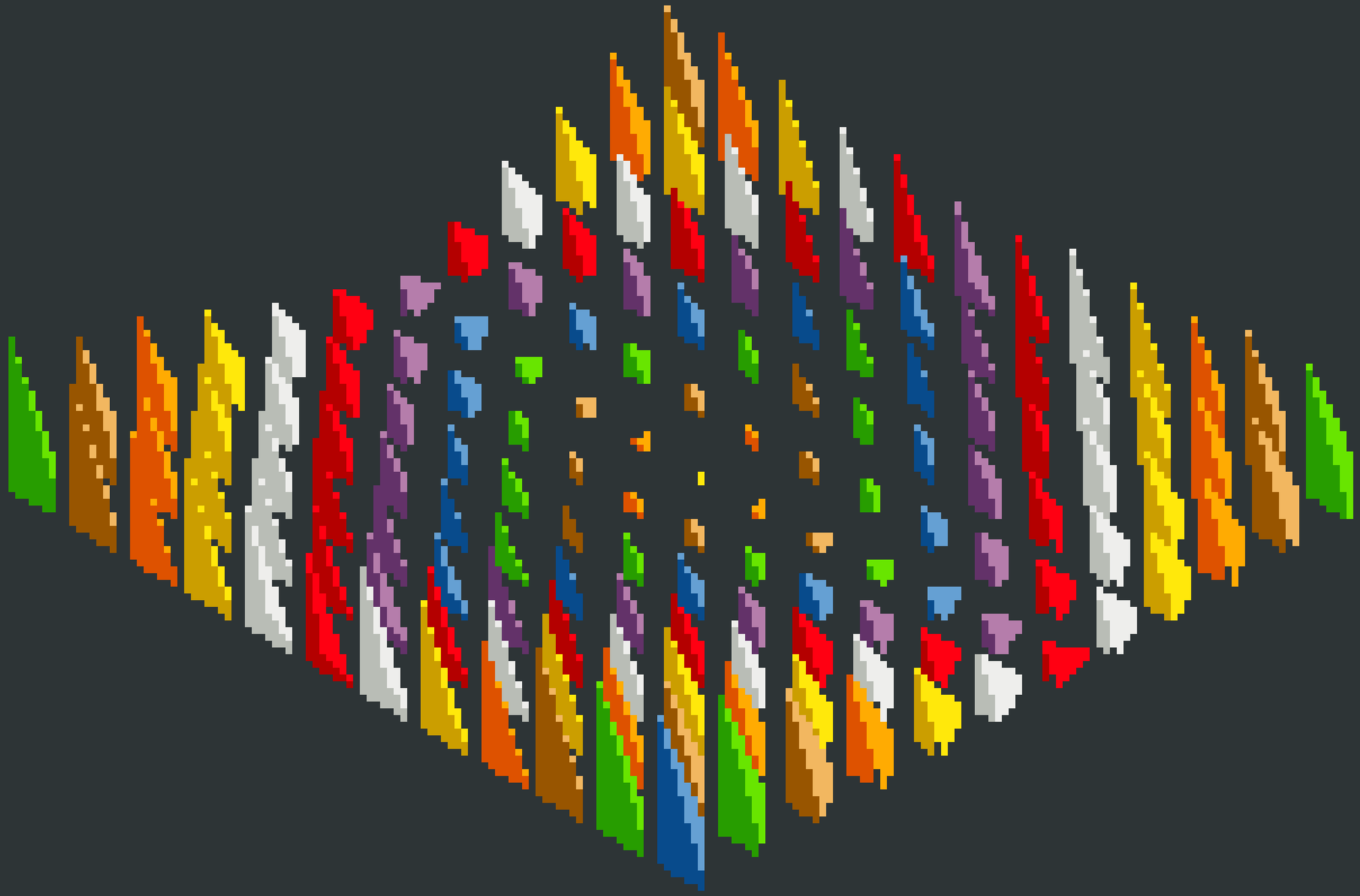
Three blocks

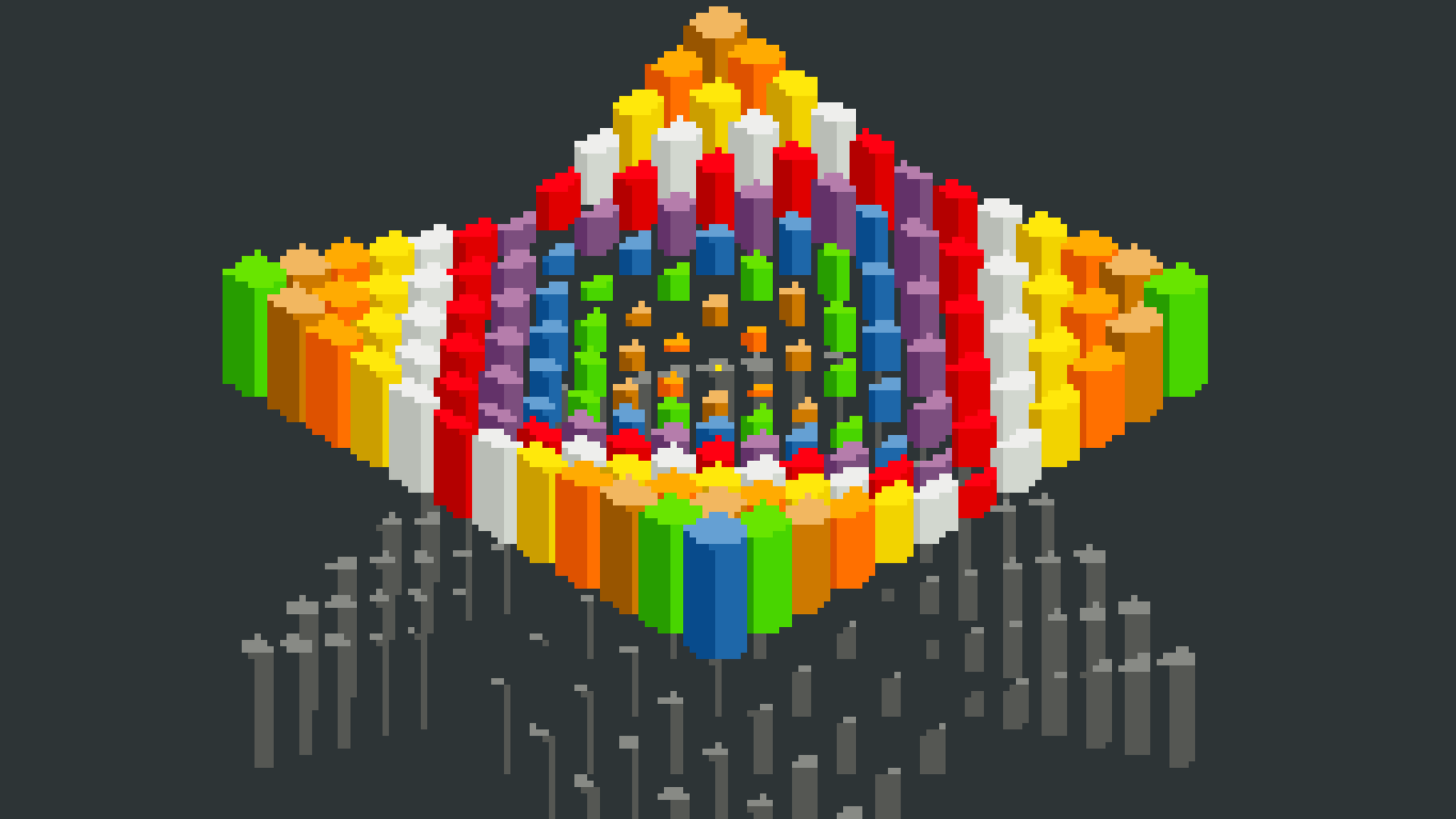
```
package main

import "github.com/peterhellberg/gfx"

func main() {
  m := gfx.NewImage(1080, 1080)
  v := gfx.V(1.2, -1.94)
  o := gfx.BoundsCenterOrigin(m.Bounds(), v, 320)
  b := gfx.Blocks{
    {gfx.V3(0, 0, 0), gfx.V3(1, 1, 1), gfx.BlockColorRed},
    {gfx.V3(0, 1, 0), gfx.V3(1, 1, 1), gfx.BlockColorGreen},
    {gfx.V3(1, 1, 0), gfx.V3(1, 1, 1), gfx.BlockColorBlue},
  }

  b.Sort()

  b.DrawPolygons(m, o)

  gfx.SavePNG("three-blocks.png", m)
}
```











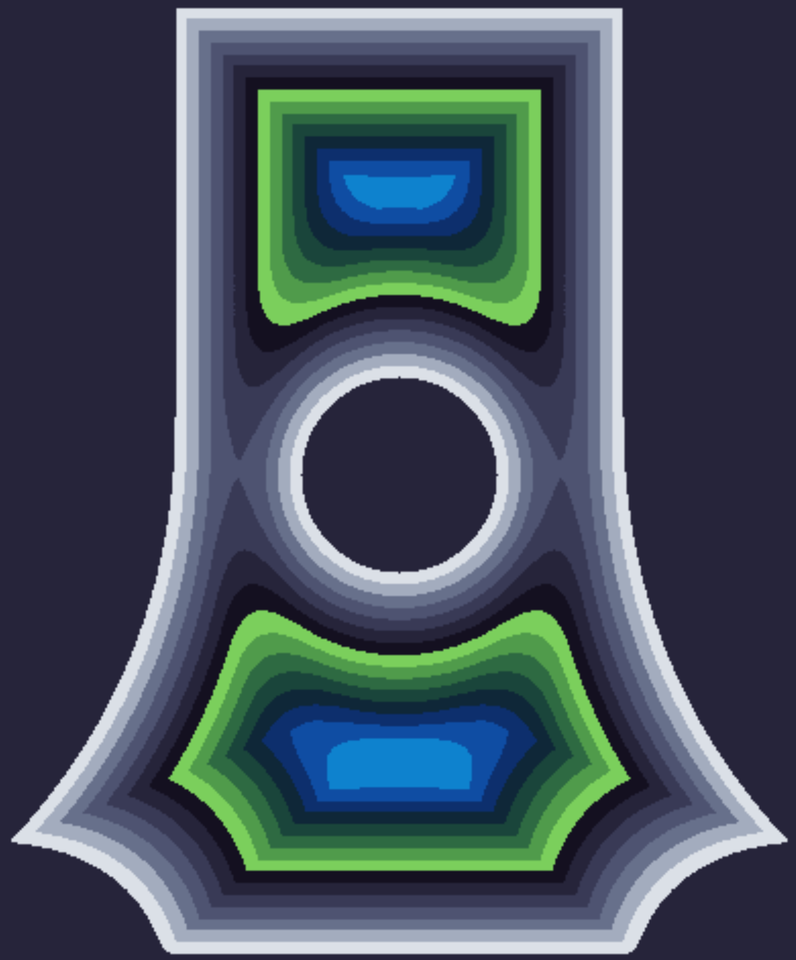
SDF

iquilezles.org/www/articles/distfunctions2d/distfunctions2d.htm

Repeat: Subtract Circle from Line

```
package main

import "github.com/peterhellberg/gfx"

func main() {
  c := gfx.PaletteEDG36.Color
  m := gfx.NewImage(2560, 1440, c(5))

  gfx.EachImageVec(m, gfx.ZV, func(u gfx.Vec) {
    sd := gfx.SignedDistance{u}

    if d := sd.OpRepeat(gfx.V(256, 256), func(sd gfx.SignedDistance) float64 {
      return sd.OpSubtraction(sd.Circle(50), sd.Line(gfx.V(0, 0), gfx.V(64, 64)))
    }); d < 40 {
      gfx.SetVec(m, u, c(int(gfx.MathAbs(d/5))))
    }
  })

  gfx.SavePNG("sdf-repeat.png", m)
}
```

EquilateralTriangle



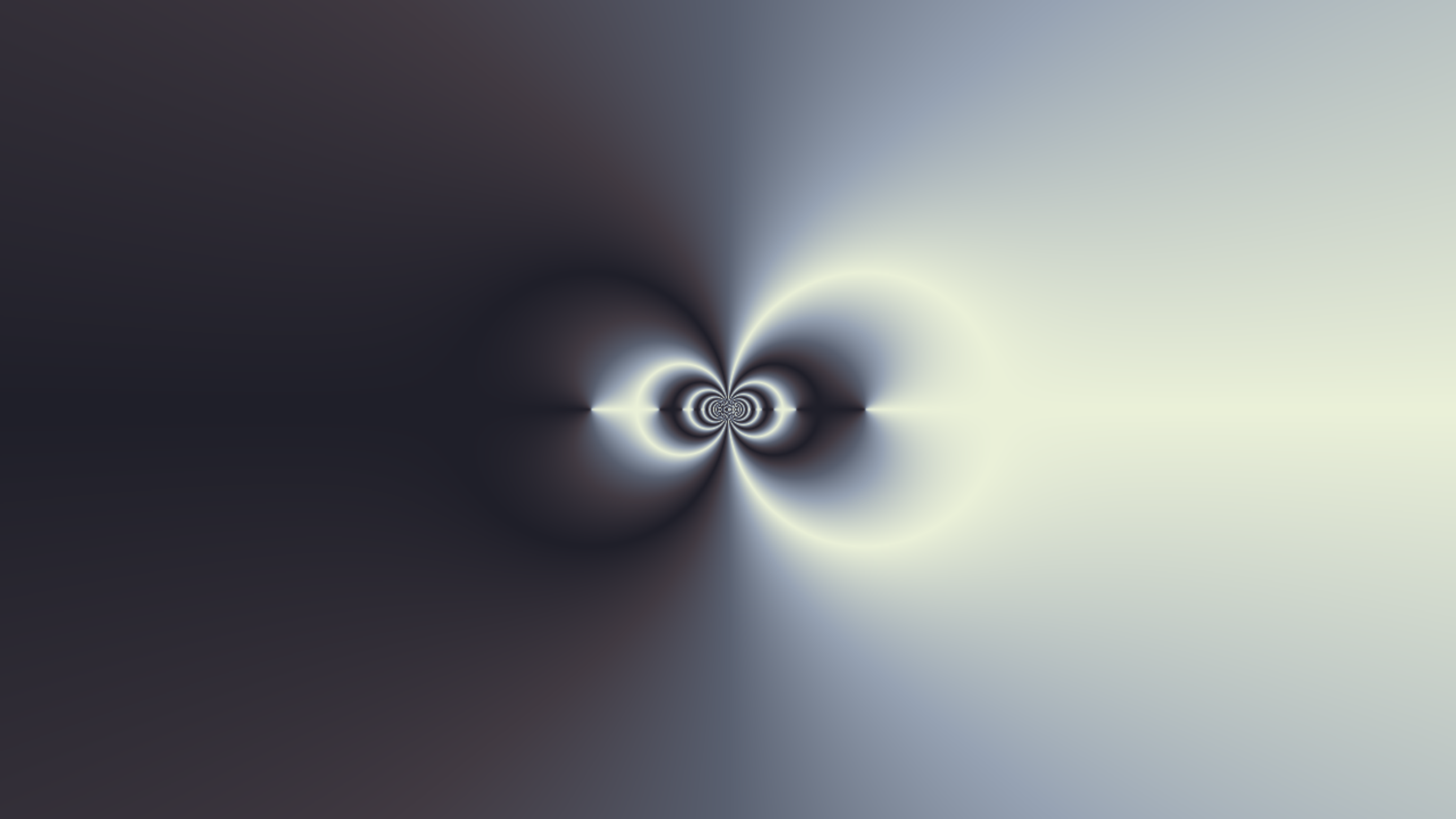
SmoothUnion of Rectangle
and IsoscelesTriangle

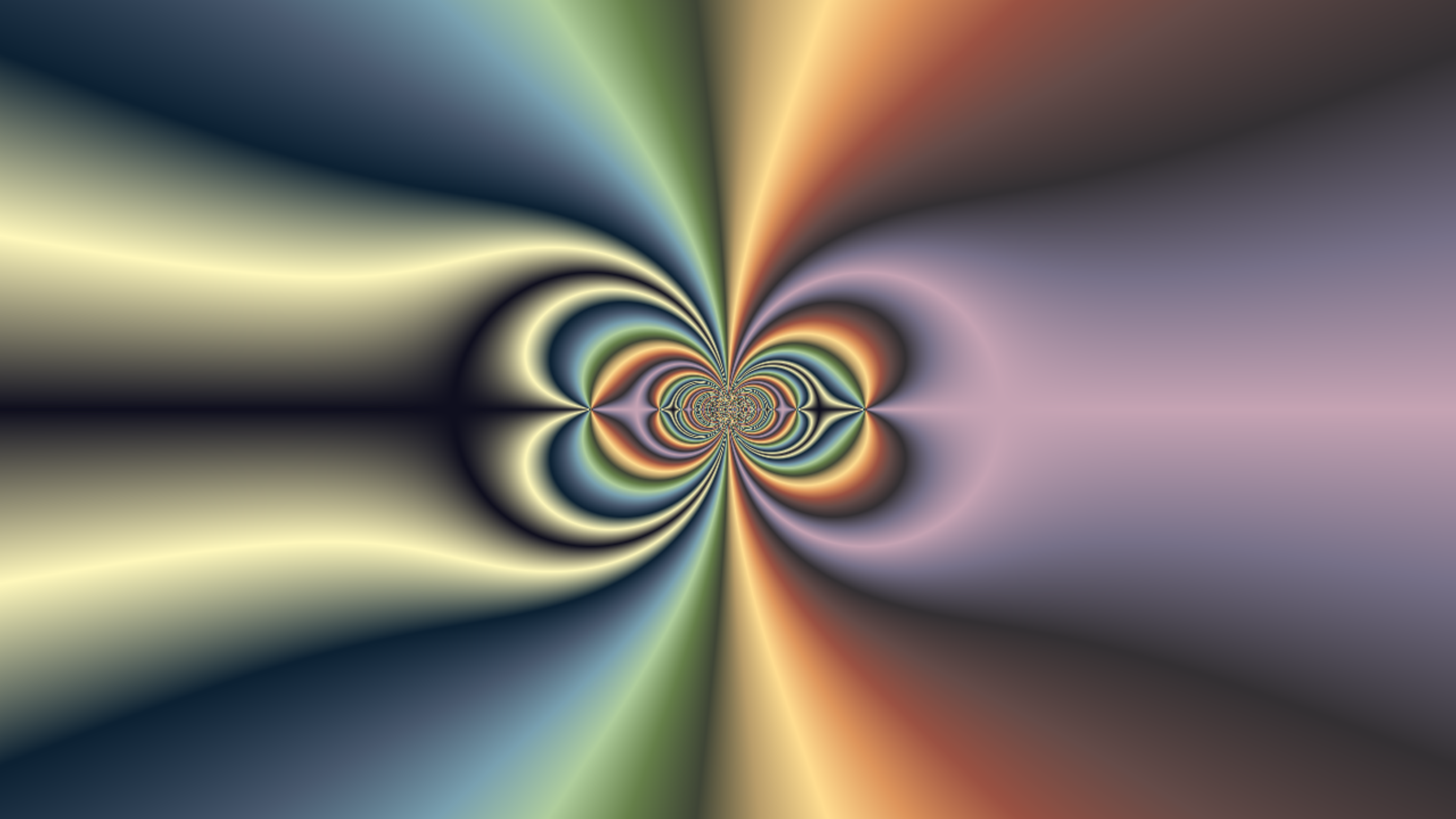
Domain coloring

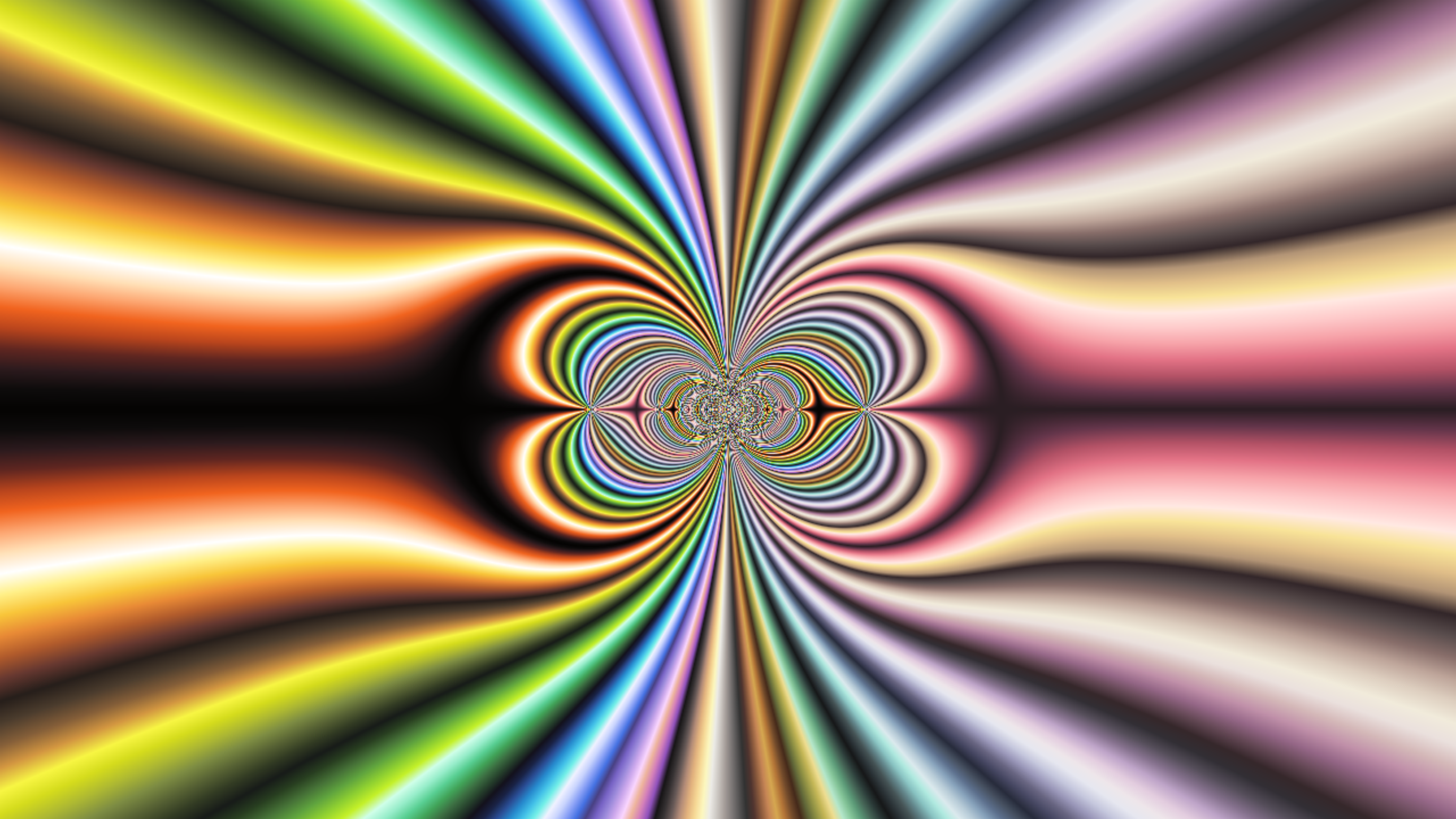
Domain coloring

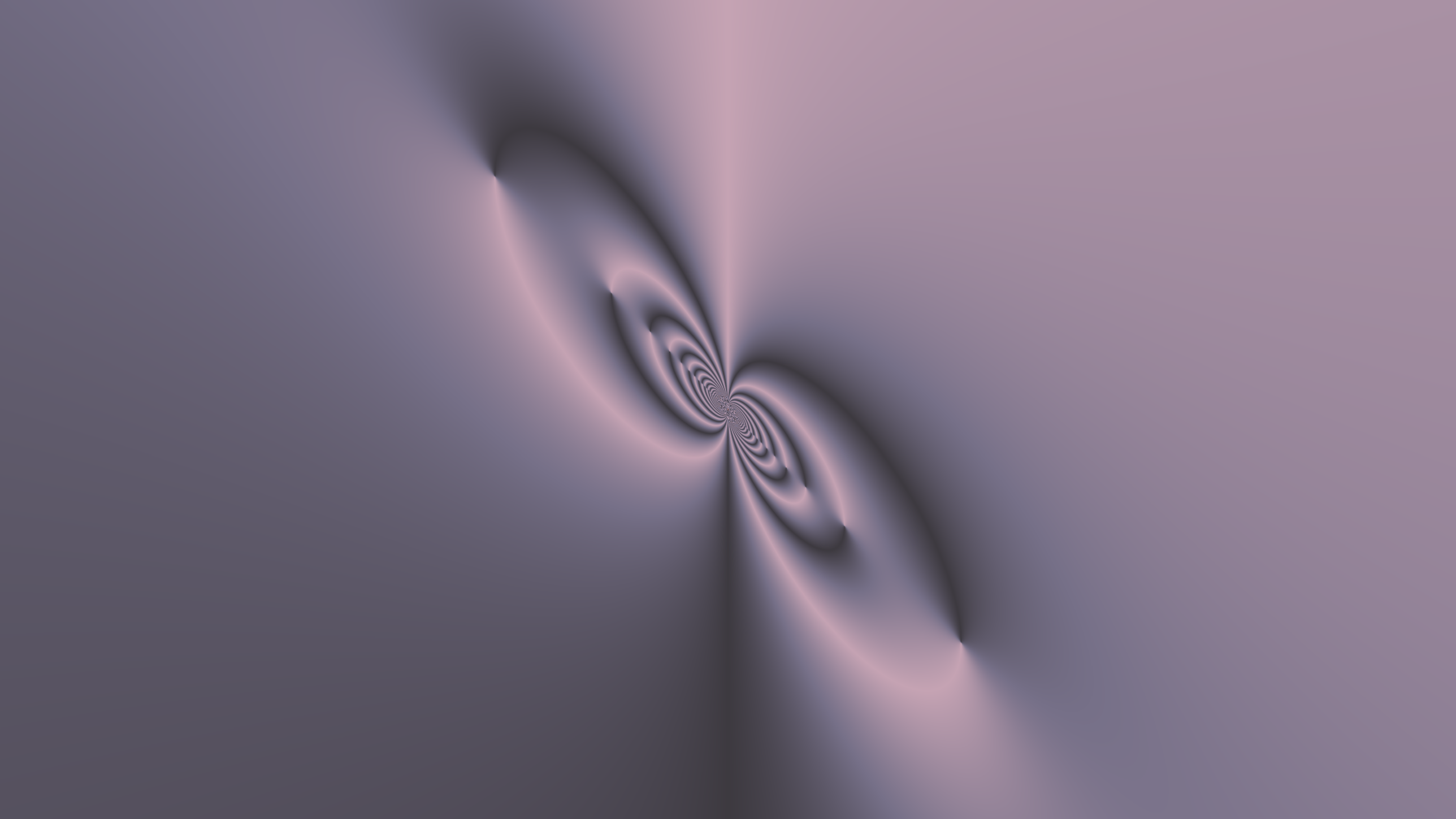
```go
package main

import "github.com/peterhellberg/gfx"

const (
  w, h, fovY = 2560, 1440, 1.9
  aspect     = float64(w) / float64(h)
  ahc        = aspect * fovY / 2.0
  hfc        = fovY / 2.0
)

func pixelCoordinates(px, py int) gfx.Vec {
  return gfx.V(
    ((float64(px)/(w-1))*2-1)*ahc,
    ((float64(h-py-1)/(h-1))*2-1)*hfc,
  )
}
```

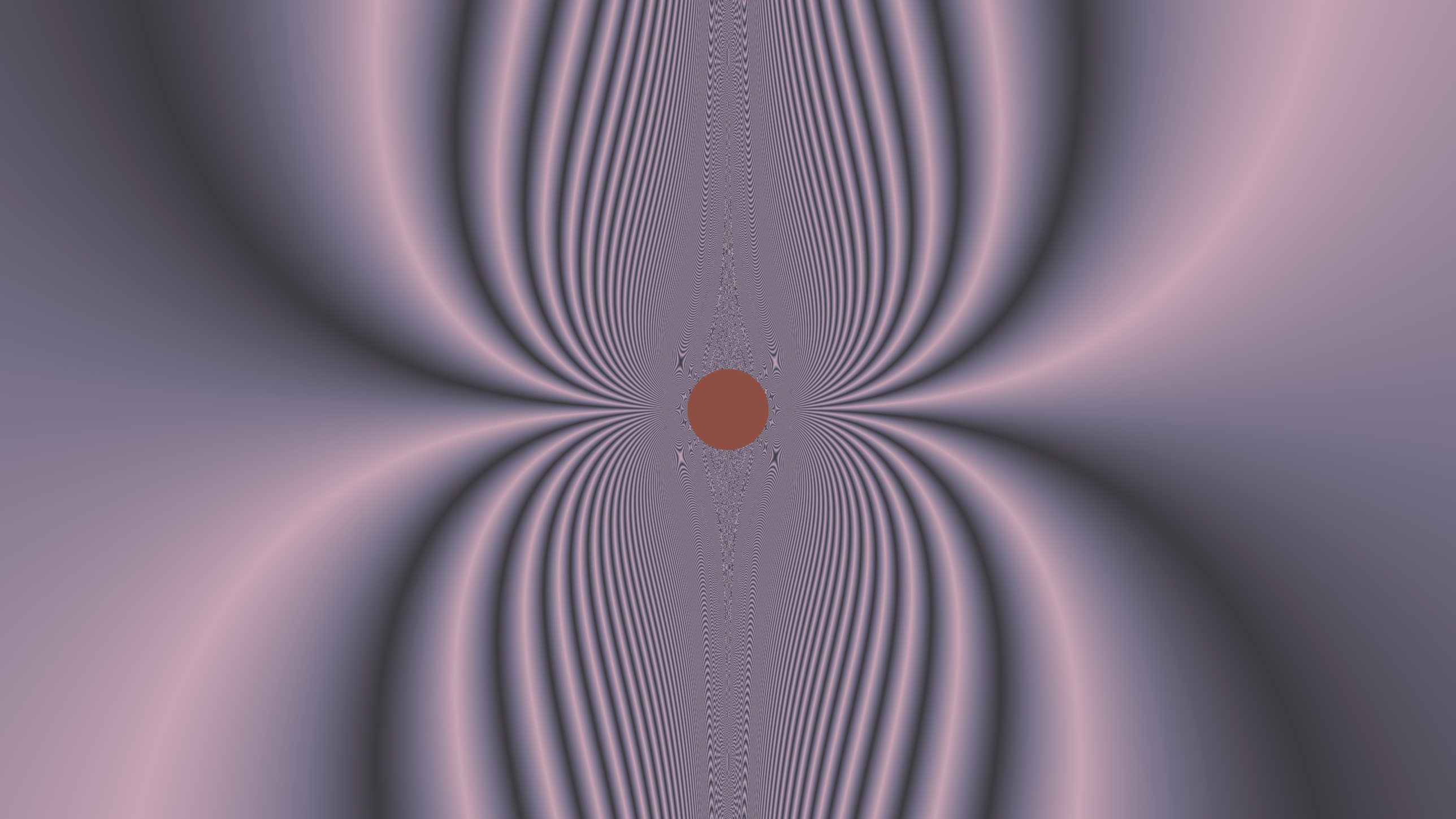
```go
func main() {
  var (
    p0 = pixelCoordinates(0, 0)
    p1 = pixelCoordinates(w-1, h-1)
    y  = p0.Y
    d  = gfx.V((p1.X-p0.X)/(w-1), (p1.Y-p0.Y)/(h-1))
    m  = gfx.NewImage(w, h)
  )

  for py := 0; py < h; py++ {
    x := p0.X

    for px := 0; px < w; px++ {
      z := gfx.CmplxSin(1 / complex(x, y))
      c := gfx.PaletteEN4.CmplxPhaseAt(z)
      m.Set(px, py, c)
      x += d.X
    }

    y += d.Y
  }

  gfx.SavePNG("domain-coloring.png", m)
}
```

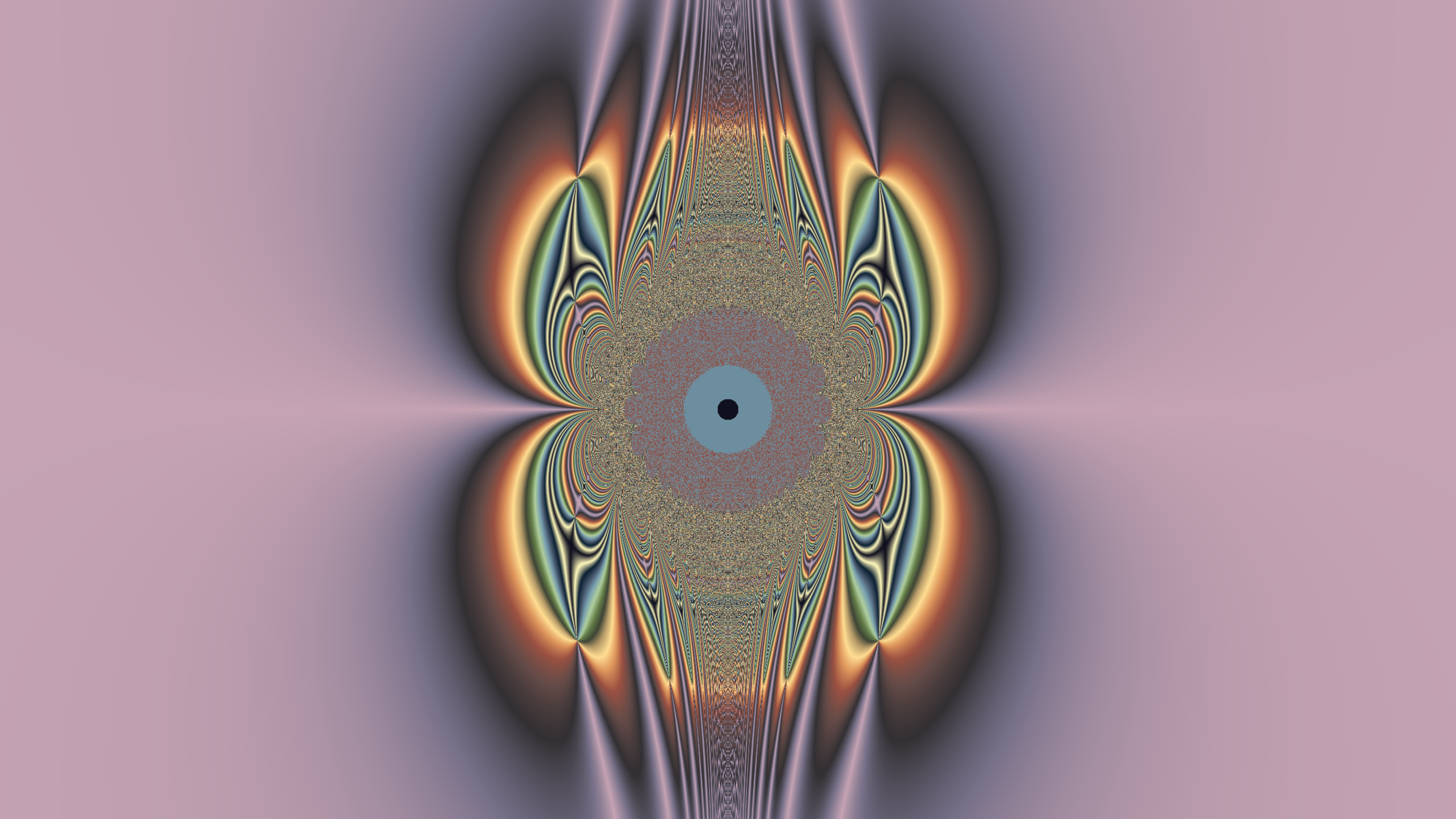


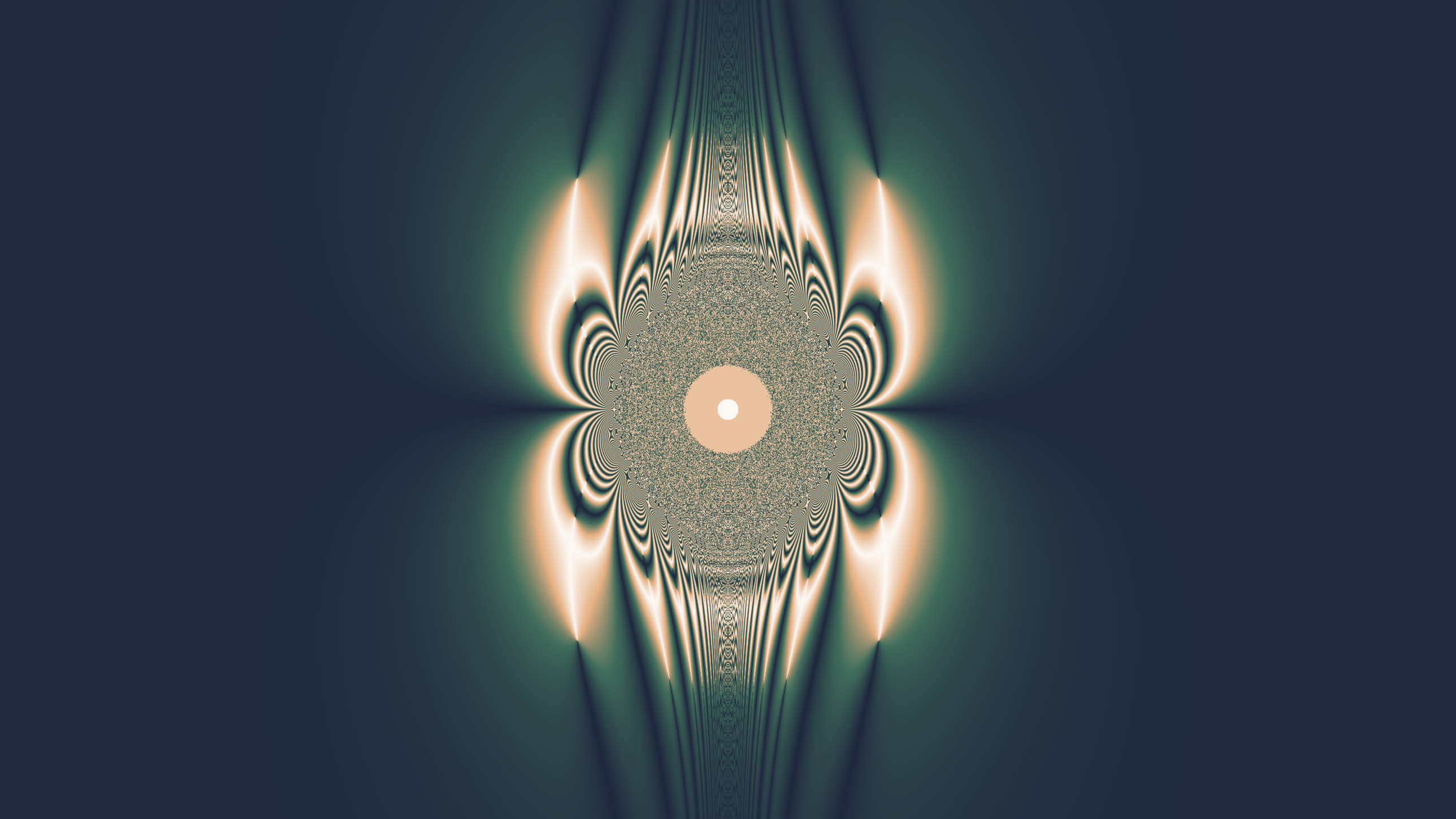


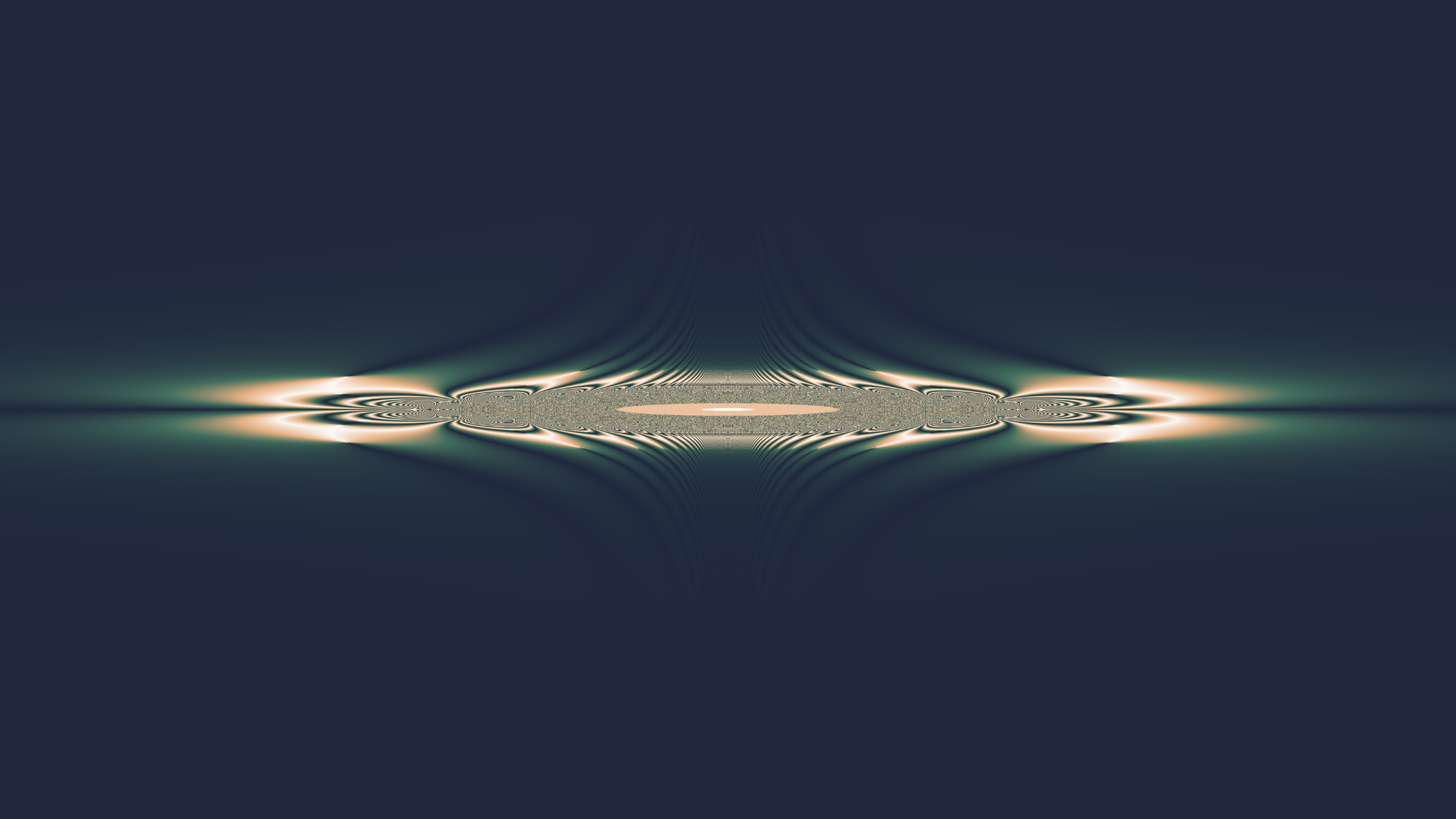


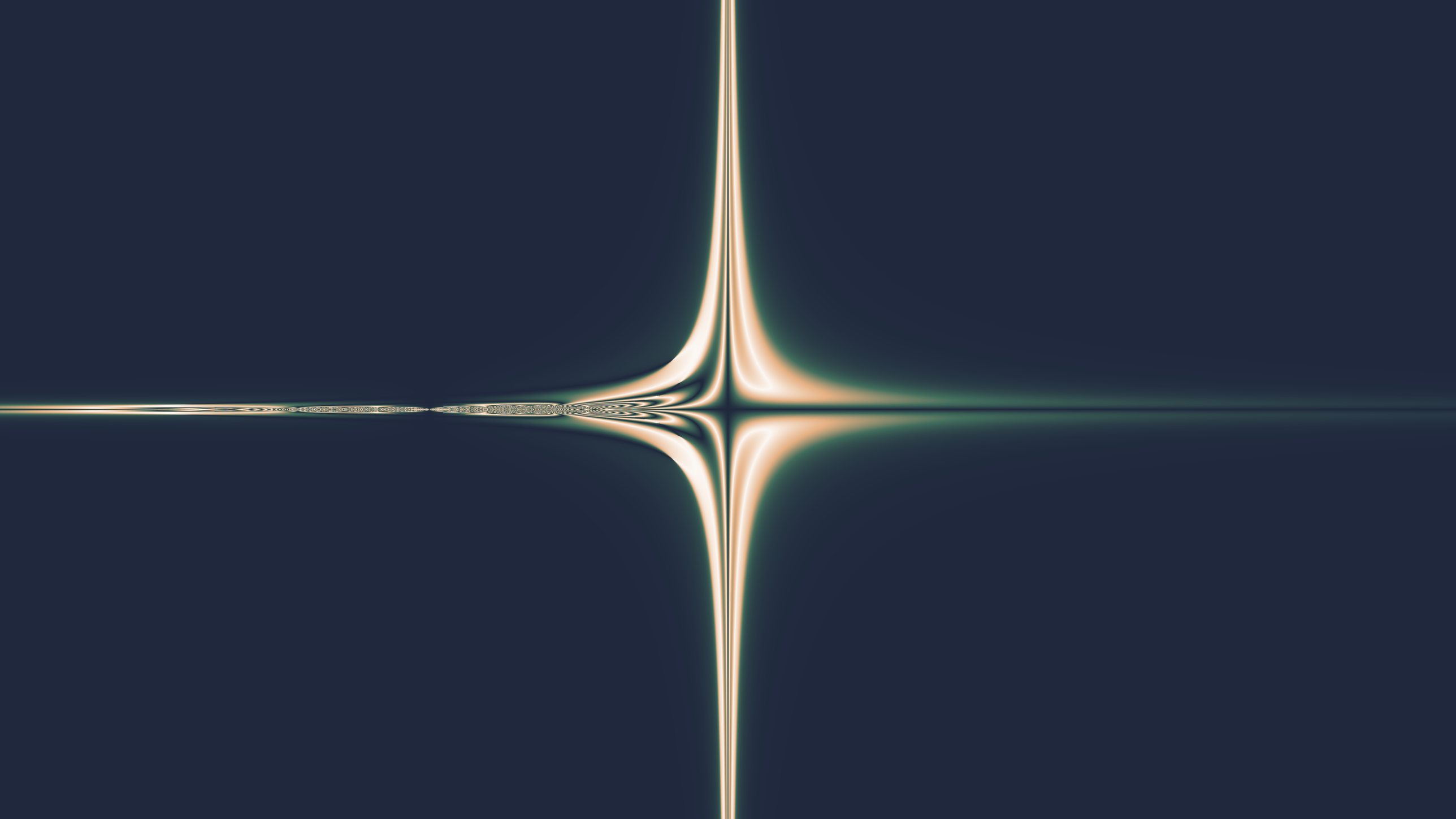


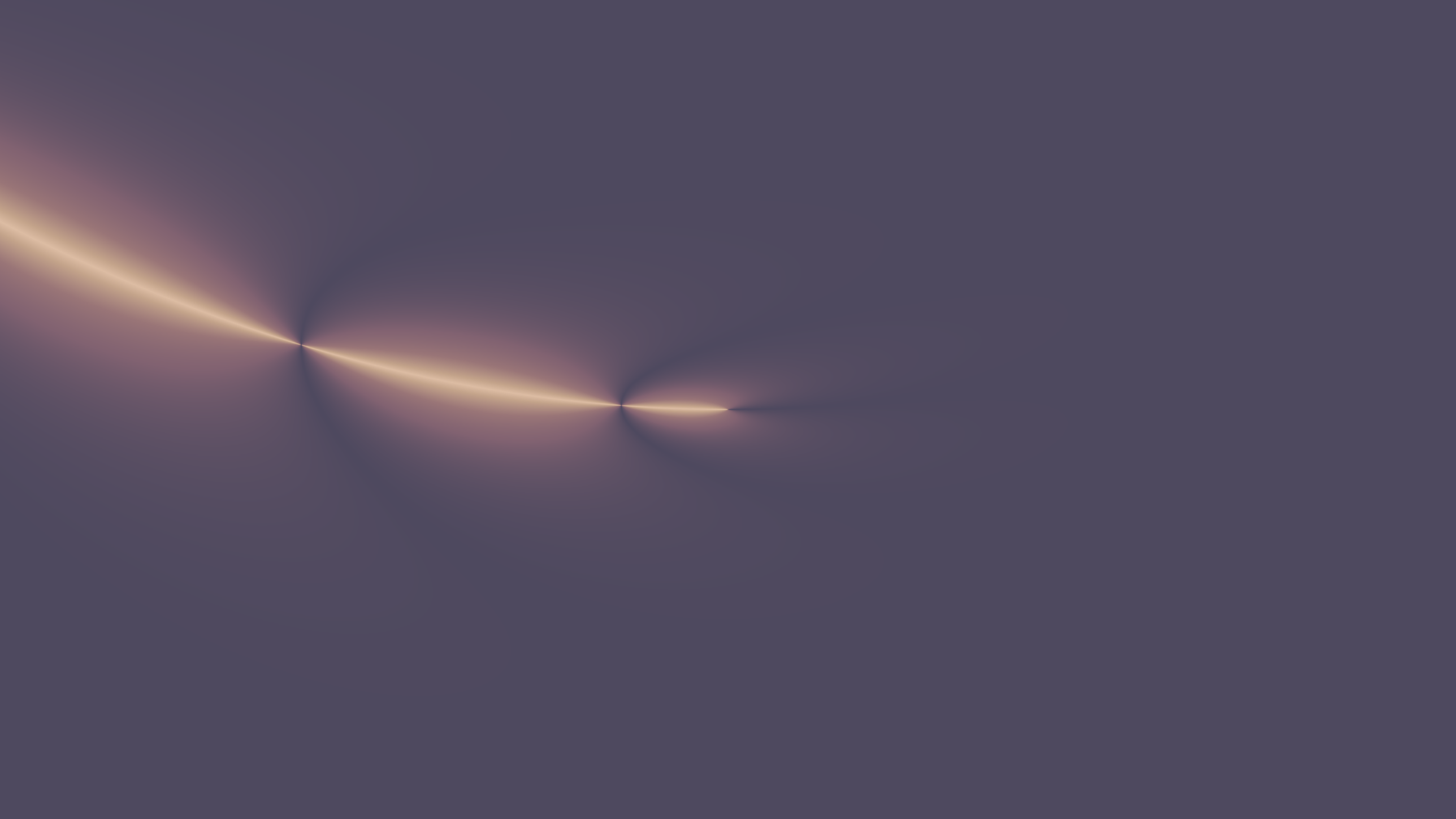


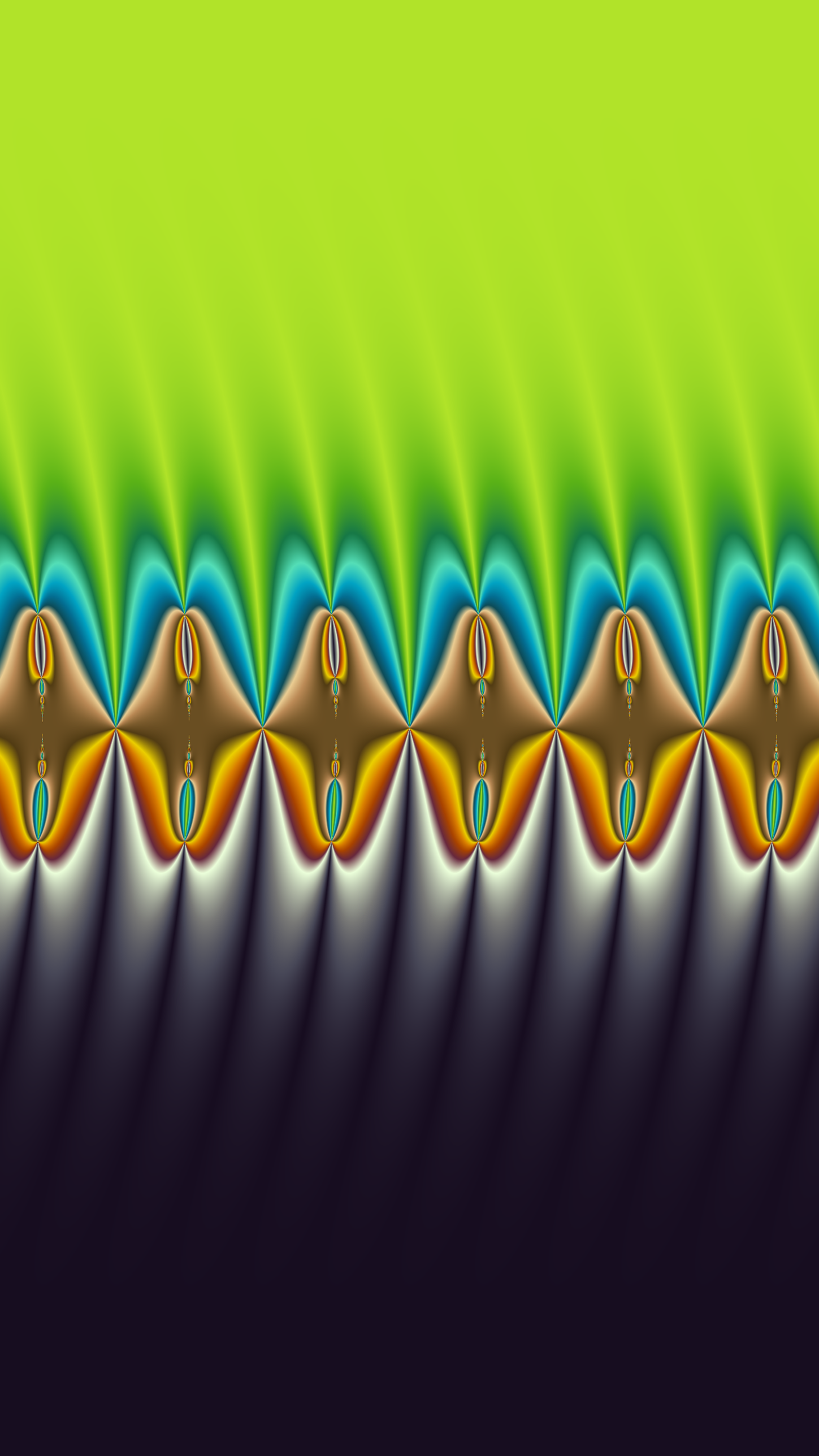




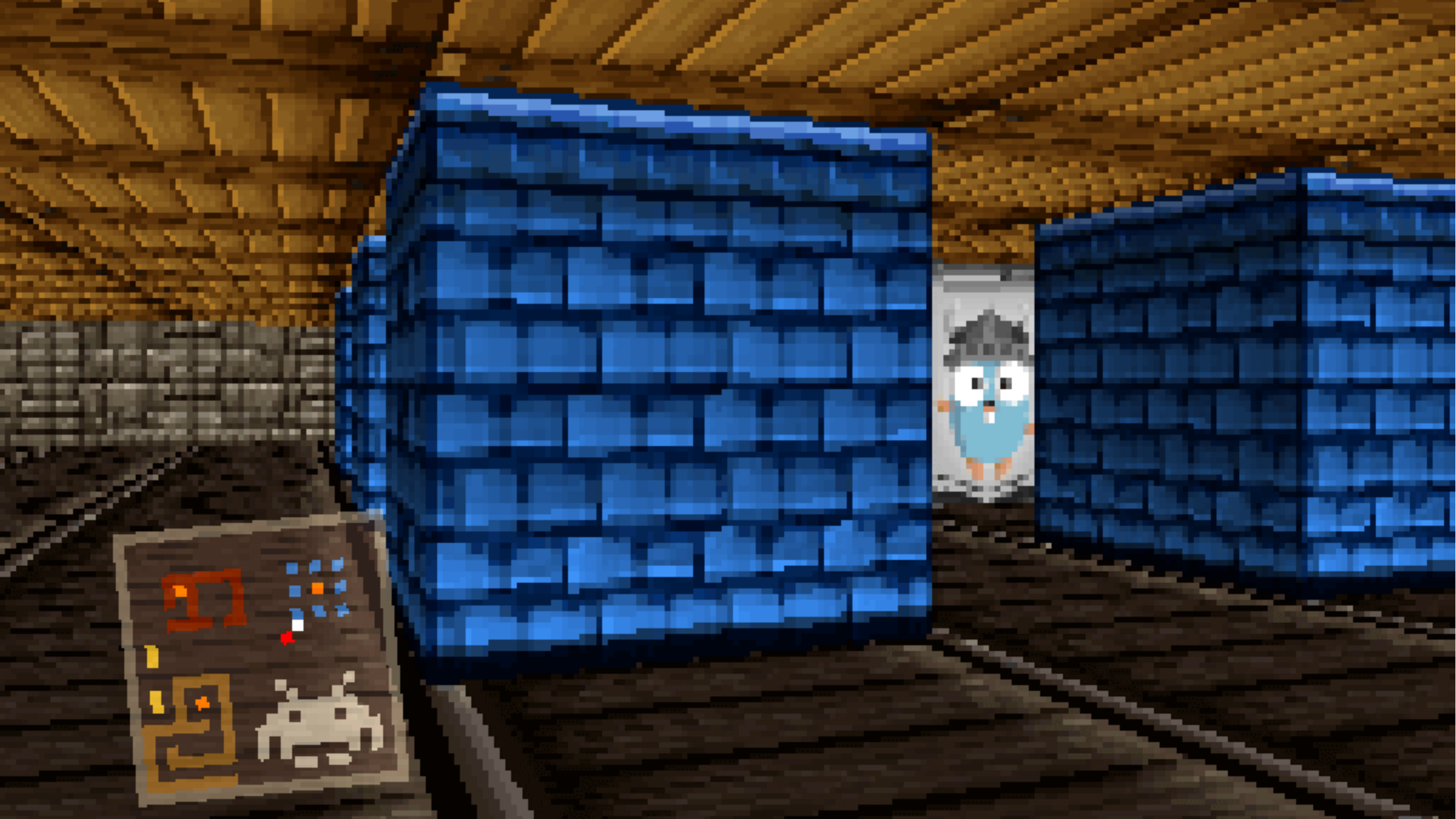


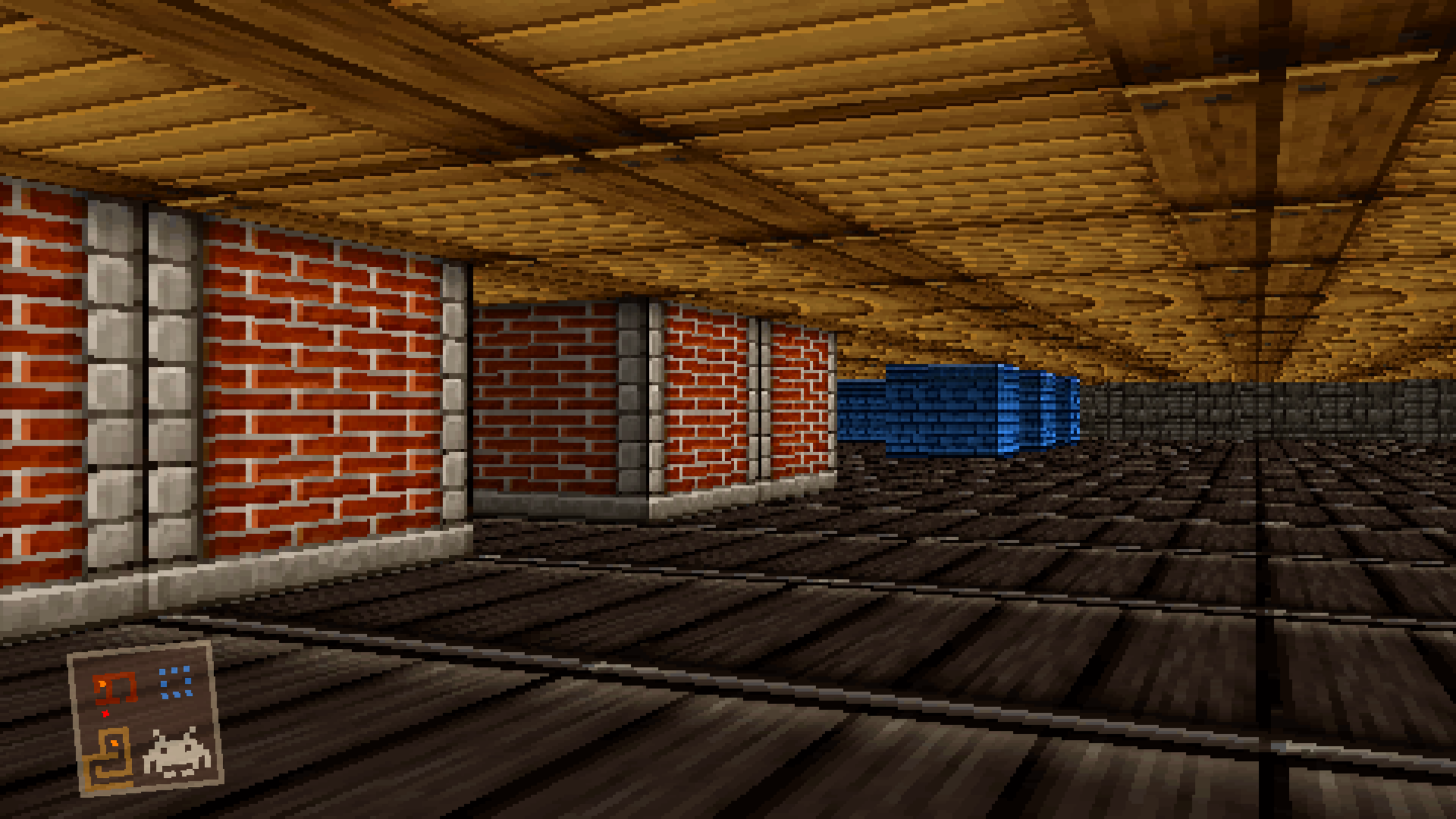


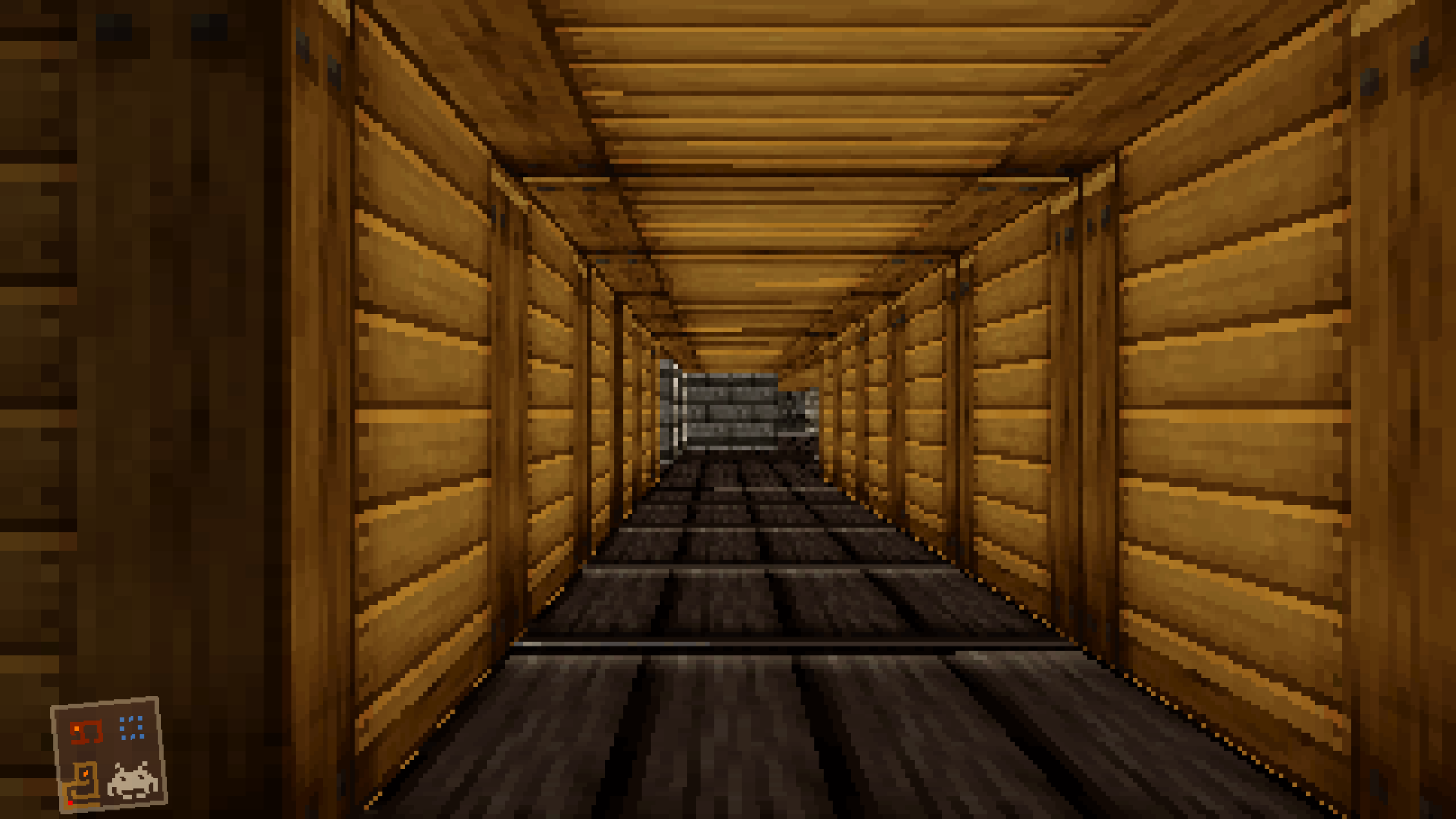




Ray Casting





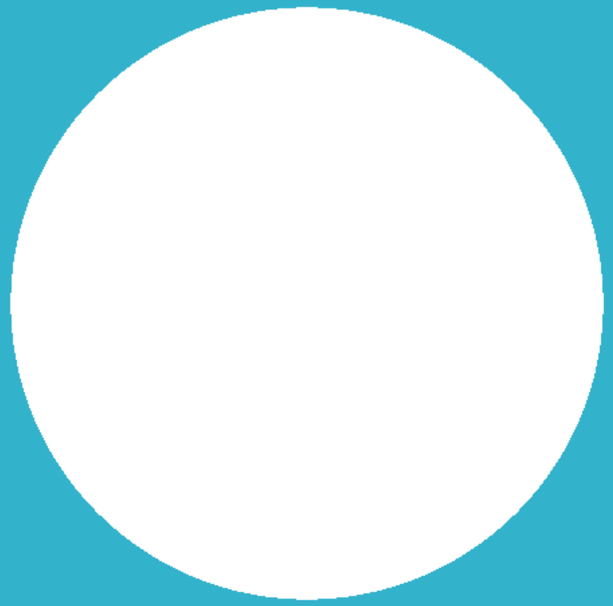


Ray Marching

KABOOM!

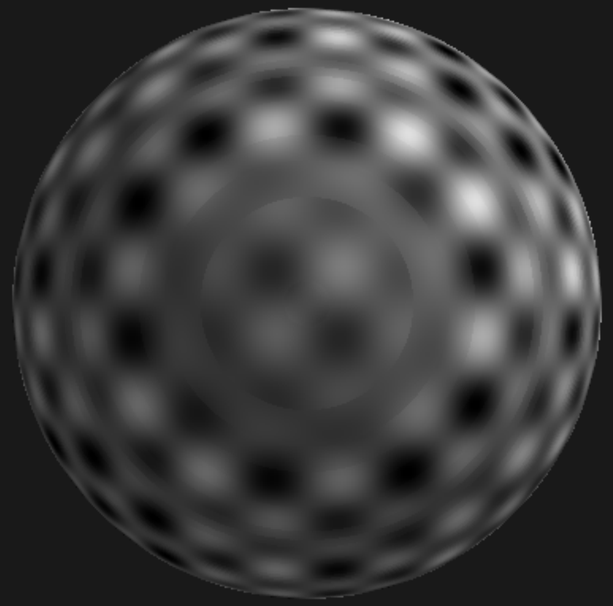
github.com/ssloy/tinykaboom

Draw a sphere

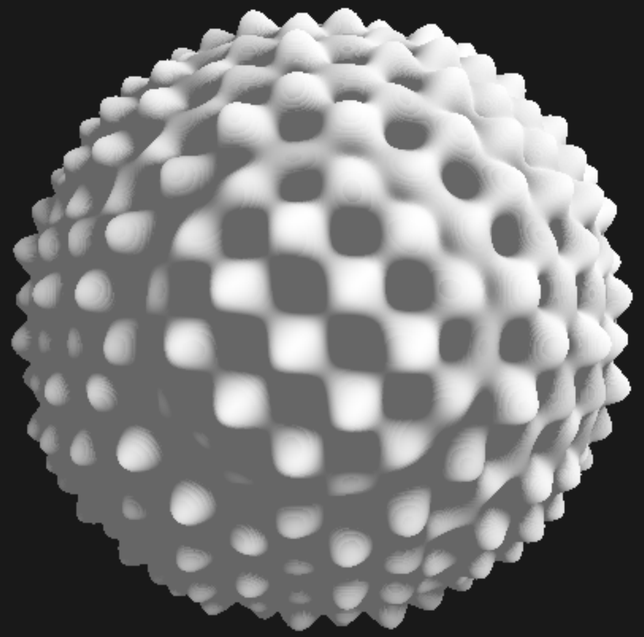


Diffuse shading

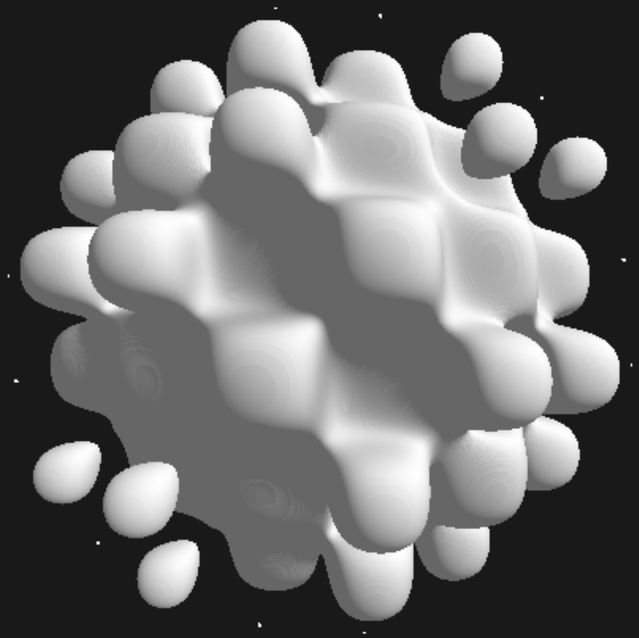
Draw a pattern



Displacement mapping

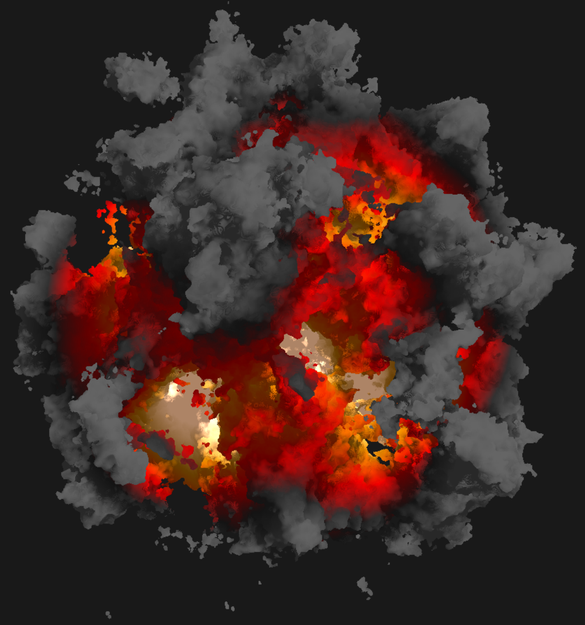


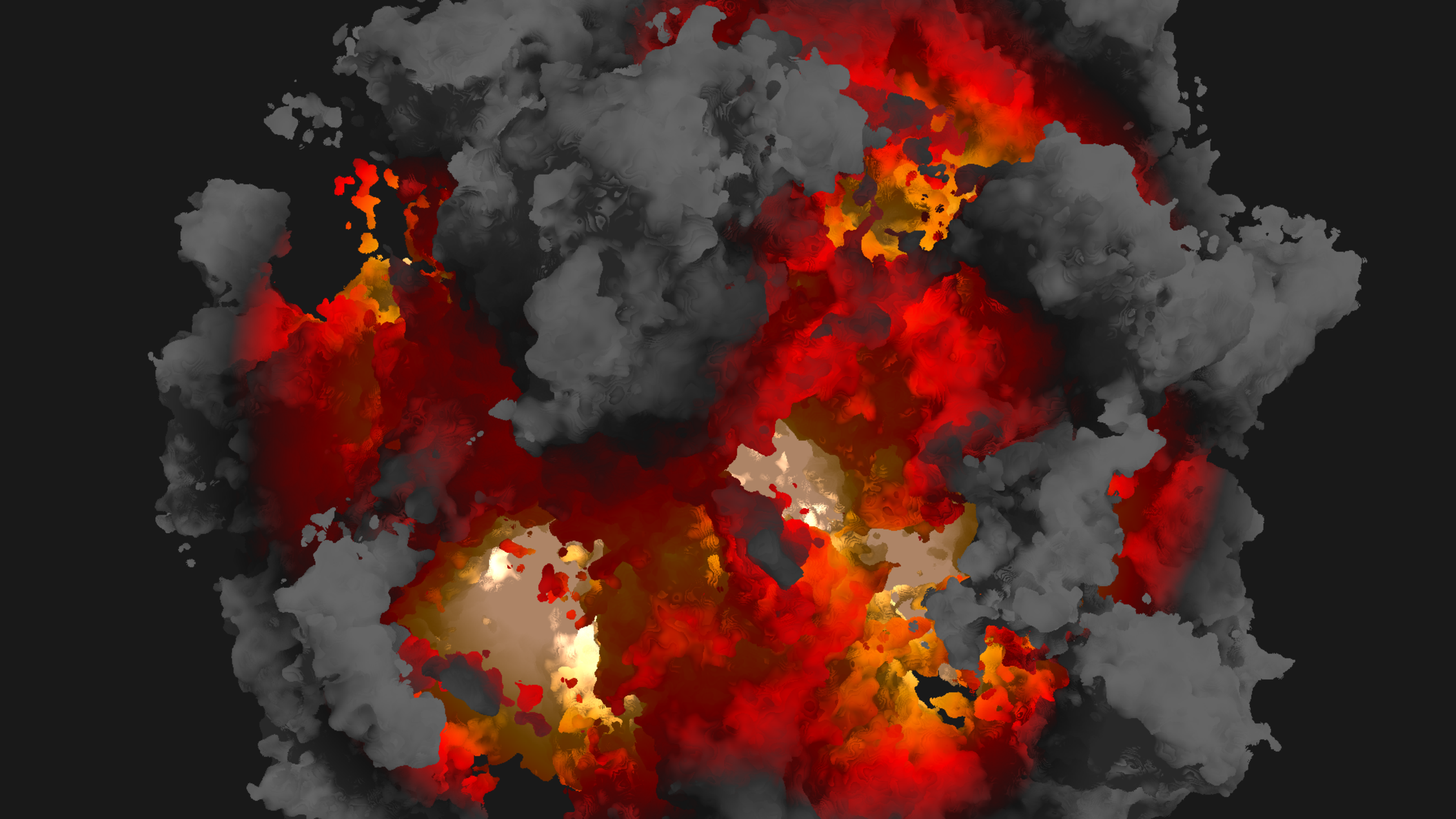
Another implicit surface



Pseudorandom Noise

Fire colors





gist.github.com/peterhellberg